AllFusion® ERwin® Data Modeler

API リファレンス ガイド

r7.2

CA 製品のリファレンス

このドキュメントでは、次の CA 製品について説明します。

- AllFusion® ERwin® Data Modeler
- AllFusion® Model Manager

テクニカル サポートへのお問い合わせ

テクニカル サポートの連絡先、および営業時間については、日揮情報システム株式会社のテクニカル サポート ページ（http://www.jsys-products.com/support/techinfo/index.html）をご参照ください。

目次

第1章　はじめに

AllFusion ERwin DM には、高度なカスタマイズ機能を提供する Script Client API が含まれています。この API を使用すると、モデリング データにアクセスして、さまざまな操作を実行することができます。操作対象となるのは、実行時にメモリに格納されたモデル、ファイルに保存されたモデル、または AllFusion Model Manager リポジトリに保存されたモデルです。API インターフェイスはオートメーション互換であるため、サードパーティの開発者やスクリプト ベース環境のユーザーは、AllFusion ERwin DM の豊富な設計機能を活用することができます。

さらに、スクリプト、アドイン、および COM ベースの API テクノロジを利用して、オリジナルのモデリング ツールにカスタム コンポーネントを追加することができます。API の柔軟な機能を活用することで、クライアントの開発サイクルにモデリング ツールをシームレスに統合することができます。

本章には次のトピックがあります。

主な機能 (10ページを参照)
基本的な活用例 (10ページを参照)

主な機能

API は、次のような機能を含むインターフェイスから構成されています。

アクティブ モデル データ オブジェクト（AMDO）

サードパーティのクライアント ソフトウェアが、COM オートメーション互換 API を通じてモデル データにアクセスできるようにします。この機能は、API の主要なコンポーネントです。API を構成するすべてのインターフェイスは、オートメーション ベースであるため、デュアル インターフェイスです。デュアル インターフェイスによって、メソッドおよびプロパティへより高速にアクセスできます。デュアル インターフェイスを使用すると、Invoke() 関数を使用せずに関数を直接呼び出すことができます。

コレクションと列挙子

AMDO オートメーション機能を使用するスクリプト言語でのプログラミングが容易になります。

接続ポイント

接続ポイント インターフェイスのコレクションと、ITypeInfo2 インターフェイスのサポートを提供し、プログラミング言語で同期イベントを実装できるようにします。

オートメーションの高度なエラー処理

API コンポーネントの IErrorInfo インターフェイスによって、オートメーションの高度なエラー処理をサポートします。

アクティブ モデル ディレクトリ

AllFusion Model Manager などのモデル ストレージを利用できるようにします。クライアントは、ファイルや AllFusion Model Manager リポジトリに保存されたモデルを開いたり、新しいモデルを作成することができます。

アクティブ スクリプト

スクリプト環境をホストして、スクリプトおよびアドイン コンポーネントの起動メカニズムを提供します。このメカニズムを使用して、アドインとスクリプトレットをアクティブ スクリプト環境に登録できます。

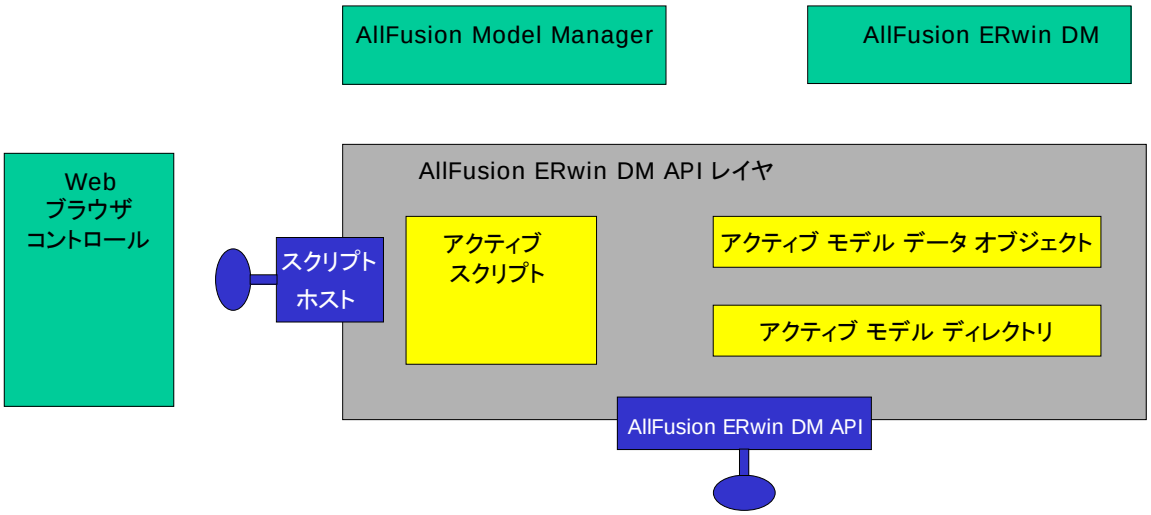
基本的な活用例

API の基本的な活用例は、オートメーションおよびスクリプトで、COM オートメーション標準に基づいた特定のインターフェイス設計要件をサポートすることです。たとえば、特定の COM オブジェクトの入出力インターフェイスが 1 つだけに制限されることがあります。このような制限が生じるのは、オートメーションのみで認識できるインターフェイス タイプは IDispatch だけであり、QueryInterface 機能の使用は不適切となるためです。この問題に対処する一般的な手法は、セカンダリ インターフェイスを表す代替 ID と読み取り専用プロパティを含めることです。

もう一つの例は、別の言語（C++以外）を使用してクライアントを実装する場合です。Visual Basic、VB Script、Java Script などの言語を使用できます。オブジェクトのコレクション、接続ポイント、高度なエラー処理など、言語と COM のバインドをカプセル化する特別な構文が含まれています。

API はさまざまなコンポーネントを組み合わせて、COM からアクセス可能なインターフェイス セットとして表します。

統合されているコンポーネントは、AllFusion ERwin DM、AllFusion MM、および Microsoft Internet Explorer です。



スタンドアロン クライアント

API は、スタンドアロン クライアントで使用することができます。サードパーティのクライアントは、API をインプロセス サーバーとしてアクティブにします。API コンポーネントには視覚的な表示機能がなく、ユーザー インターフェイスが表示されません。API にはアクティブ モデル ディレクトリ機能があり、利用可能なモデルの一覧から対象モデルを指定することができます。アクティブ モデル データ オブジェクトを使用すると、モデル データにセッション ベースでアクセスできます。

API クライアントがモデル データにアクセスする際は、そのモデルを使用している他のユーザーと競合する可能性があります。AllFusion MM を使用すると、高度なモデル共有機能によって、セッション中に他のユーザーがモデルにアクセスするのを防止できます。

アドイン コンポーネントまたはスクリプト

API は、アドイン コンポーネントまたはスクリプトで使用することもできます。AllFusion ERwin DM は、サードパーティのアドイン モジュールやスクリプトをホストすることができます。API のアクティブ スクリプト コンポーネントを使用して、ホスト ツールへのモジュール登録、ホストのユーザー インターフェイスの表示設定、アドイン メニューの作成、およびホスト メニューの選択またはイベントによる呼び出しを行うことができます。

アドイン モジュールはクライアント DLL です。インプロセスでアクティブになります。

スクリプトは VBScript または JScript のプロシージャで、DHTML ドキュメントに埋め込まれます。メニュー項目を選択するか、またはモデル イベントによってアクティブになります。このアクティブ スクリプトは Web ブラウザの制御をホストして、DHTML オブジェクト モデルの window.external プロパティを通じて API オブジェクトを利用できるようにします。

モデルの変更点は画面上で確認することができます。また、一時停止して、モデリング ツールのユーザー インターフェイスにアクセスし、モデルの状態を確認することができます。

第2章　API コンポーネント

本章には次のトピックがあります。

概要

API は、AllFusion ERwin DM の機能を表すインターフェイスの集合体です。アプリケーションはトップレベルのインターフェイスをエクスポートし、クライアントはそこから必要に応じて低レベルのインターフェイスを取得します。インターフェイスは複数の論理層にグループ化されており、各層にはアプリケーションの機能を表すインターフェイスが含まれています。以降のセクションで各層について説明します。層ごとにグループ化されたインターフェイスの説明も表に示します。

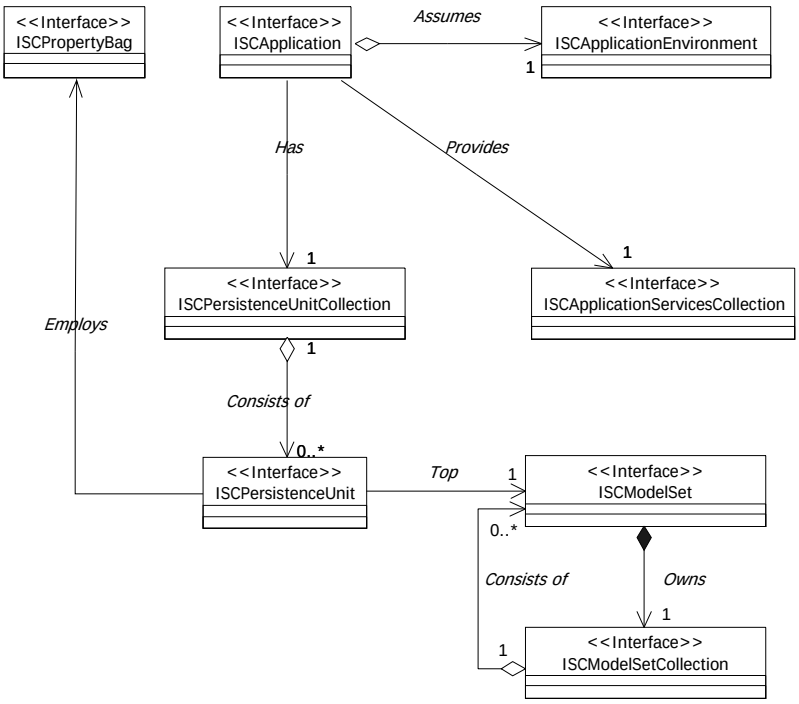
アプリケーション層

アプリケーション層は AllFusion ERwin DM の機能を表しており、永続ストレージ内のモデルへのアクセスを確立します。また、メモリ内のモデルと永続ストレージ内のモデルとの間の切り替えを制御します。次の表に、アプリケーション層のインターフェイスを説明します。

インターフェイス	役割
ISCApplication	アプリケーション全体の機能を表し、API のインターフェイス階層のエントリ ポイントになります。利用可能な永続ユニットの一覧と、クライアントと永続ユニット間の接続を保持します。
ISCApplicationEnvironment	ランタイム環境についての情報を提供します。
ISCApplicationServiceCollection	さまざまなアプリケーション サービスへのアクセスを提供します（たとえば、フォワード エンジニアリング、リバース エンジニアリング、完全比較など）。
ISCPersistenceUnitCollection	アプリケーションで認識されたすべてのアクティブな永続ユニットを収集します。
ISCPersistenceUnit	アプリケーション内のアクティブな永続ユニット（AllFusion ERwin DM モデルなど）を表します。永続ユニットはモデル セットの形式でデータをグループ化します。クライアントは永続ユニットに接続して、永続ユニットおよびそのデータを操作できます。
ISCModelSetCollection	永続ユニットに関連付けられたモデル セットを表します。
ISCModelSet	1 つの永続ユニット内のモデル セットを表します（モデル データの EMX または EM2 クラスなど）。

インターフェイス	役割
ISCPropertyBag	アプリケーション層のインターフェイス コールに対する、プロパティの配列を表します。

次の図は、アプリケーション層の関係を視覚的に表したものです。

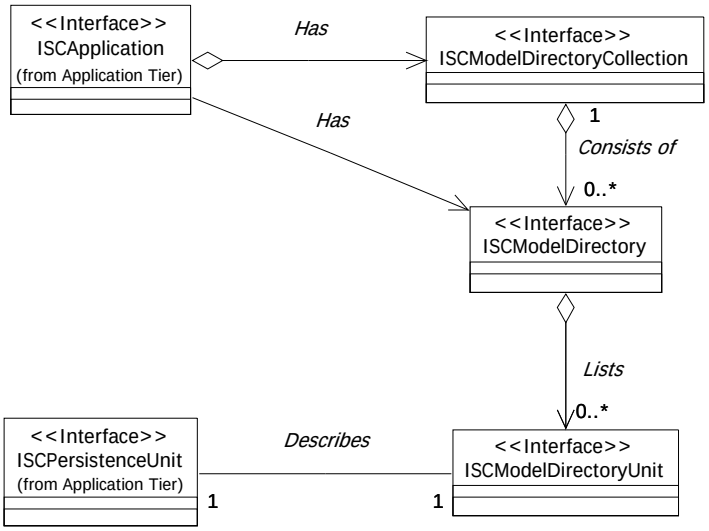


モデル ディレクトリ層

モデル ディレクトリ層は、永続ストレージ ディレクトリ（ファイル システム ディレクトリや AllFusion MM ディレクトリなど）へのアクセスおよび操作を行います。次の表に、モデル ディレクトリ層のインターフェイスを説明します。

インターフェイス	役割
ISCModelDirectoryCollection	API クライアントで利用可能なすべてのトップレベル モデル ディレクトリを列挙します。
ISCModelDirectory	1 つのモデル ディレクトリ エントリの情報をカプセル化します。
ISCModelDirectoryUnit	1 つのディレクトリ ユニットの情報をカプセル化します。

次の図は、モデル ディレクトリ層の関係を視覚的に表したものです。

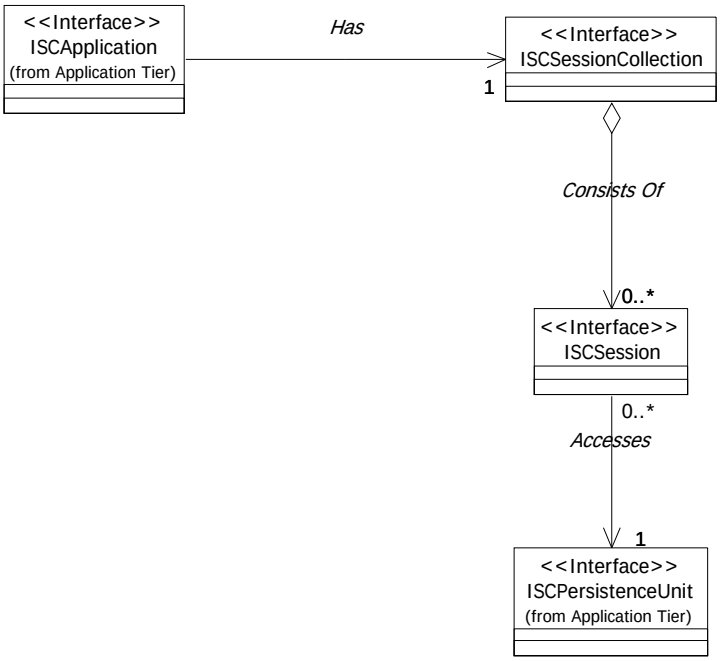


セッション層

セッション層は、メモリ内のモデル データへのアクセスを確立します。次の表に、セッション層のインターフェイスを説明します。

インターフェイス	役割
ISCSessionCollection	API クライアントと永続ユニット間のすべてのアクティブなセッションを収集します。
ISCSession	クライアントとモデル間のアクティブな接続を表します。クライアントはセッションを作成し、次に永続ユニットのモデル セットに対してセッションを開きます。開いているセッションは、モデル セットの単一レベル（データ、メタデータなど）を表します。

次の図は、セッション層の関係を視覚的に表したものです。

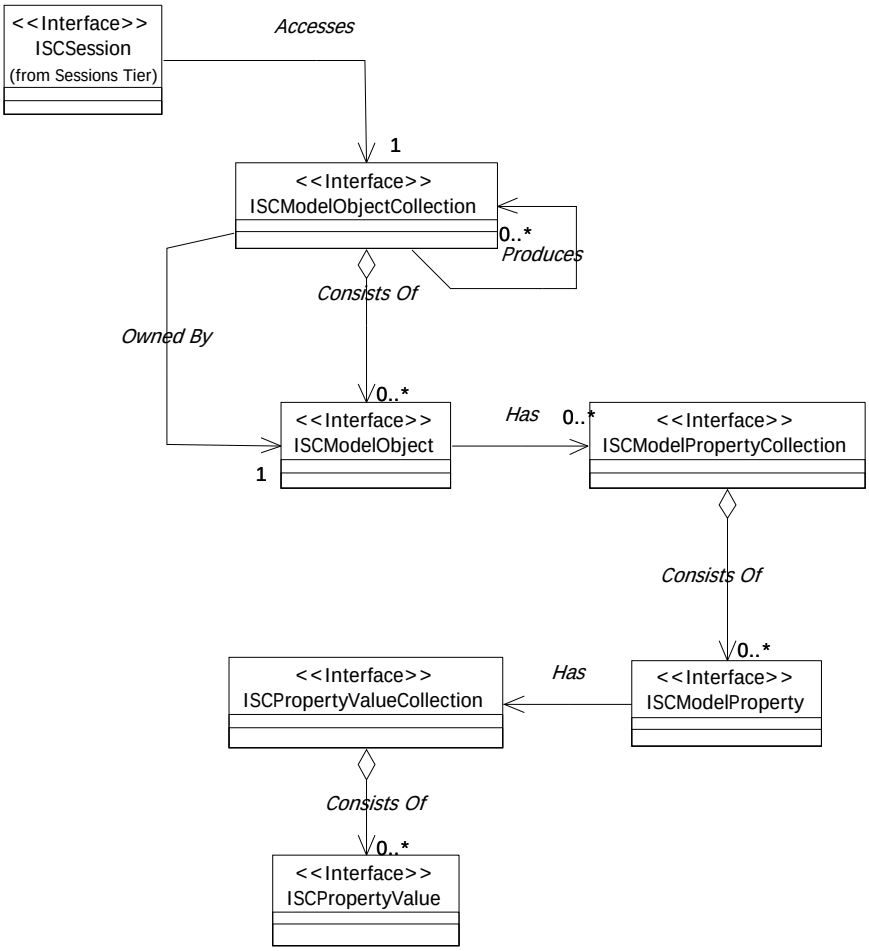


モデル データ層

モデル データ層は、モデル データへのアクセスおよび操作を行います。次の表に、モデル データ層のインターフェイスを説明します。

インターフェイス	役割
ISCModelObjectCollection	操作可能なオブジェクトを表します。このコレクションのメンバは、フィルタ条件を設定して制限することができます。
ISCModelObject	モデル内の 1 つのオブジェクトへのアクセスおよび操作を行います。
ISCModelPropertyCollection	1 つのオブジェクトが所有するプロパティの一覧を表します。この一覧はフィルタを使用して制限することができます。
ISCModelProperty	1 つのプロパティへのアクセスおよび操作を行います。プロパティは複数値を含むことがあります。複数値を含むプロパティの値にアクセスするには、キーを使用します。現在の複数値プロパティの実装では、値一覧を配列として扱い、キーは配列インデックスになります。
ISCPropertyValueCollection	1 つのプロパティの値一覧を表します。
ISCPropertyValue	1 つの値に含まれるデータおよびキーを表します。

次の図は、モデル データ層の関係を視覚的に表したものです。



モデル データへのアクセス

APIを使用して、APIクライアントからモデルを操作することができます。APIクライアントから永続ストレージ内のモデルを検索するには、Model Directory Collection、Model Directory、および Model Directory Unit コンポーネントを使用します。Model Directory Unit のプロパティを使用すると、Persistence Units コレクションを使用して、利用可能な永続ユニットのプールにユニットを登録するために必要な情報を取得できます。その後、API クライアントは読み取り専用やロック無視などのアクセス属性を指定できます。新規モデルを作成して、永続ユニット コレクションに登録できます。AllFusion ERwin DM は、ユーザー インターフェイスのアクションに応答して、プールにモデルを追加したり、プールからモデルを削除することができます。

永続ユニットが保持する一連のプロパティは、アプリケーションのユーザー インターフェイスの表示設定やアクセス属性などを制御します。永続ユニットは、モデル セットがリンクしたグループとしてデータを編成します。モデル セットはツリー型の階層構造であり、階層のトップには 1 つのモデル セットが配置されます。永続ユニットのトップ モデル セットには、モデリング データの大部分が含まれています。API では、EMX という略語を使用してトップ モデル セットを表します。EMX モデル セットにはセカンダリ モデル セットがあります。これは EM2 という略語で表され、ユーザー オプションとユーザー インターフェイス設定が含まれています。

API クライアントからモデル データにアクセスするには、セッション コンポーネントを使用して、セッションを構築し、そのセッションをモデル セットに接続します。モデル セットには、さまざまなレベルのデータが含まれています。アプリケーションが操作するデータが含まれています。エンティティ インスタンス、属性インスタンス、またはリレーションシップ インスタンスなどです。

モデル セットにはメタデータも含まれています。メタデータとは、アプリケーション データ内に配置できるオブジェクトおよびプロパティを記述したものです。AllFusion ERwin DM のメタデータには、オブジェクトとプロパティのクラス、オブジェクト集約、およびプロパティ関連が含まれています。メタデータは、モデルに配置できる各オブジェクト クラスを定義します。たとえば、エンティティ クラス、属性クラス、またはリレーションシップ クラスなどです。オブジェクト集約は、オブジェクト クラス間の所有権の関係を表します。たとえば、「モデルはエンティティを所有する」、「エンティティは属性を所有する」などの関係です。プロパティ関連は、オブジェクト クラスによるプロパティの使用を定義します。たとえば、メタデータには、Name プロパティを持つすべてのオブジェクト クラスに対するプロパティ関連が含まれています。

クライアントは、セッションをモデル セットに接続すると同時に、モデル データに必要なレベルを指定します。新規モデルが作成されると、デフォルトのオブジェクト セットが取得されます。たとえば、モデル オブジェクト、メイン サブジェクト エリア、およびストアド ディスプレイなどです。初期の API 実装では、次のレベルをサポートします。

名前	説明	サポートするアクション
SCD_SL_ M0	モデル レベル	モデル データへのアクセス、オブジェクトの作成と削除（モデル全体を含む）、プロパティ値の設定
SCD_SL_ M1	メタモデル レベル	オブジェクト定義とプロパティ定義（および他のメタデータ）へのアクセス、ユーザー定義プロパティとユーザー定義オブジェクト定義の作成と削除

レベルの識別には Long 型の整数値を使用します。値はシンボル定義を持ちます。

オブジェクトとプロパティ

API ではオブジェクト/プロパティ形式でデータを表します。AllFusion ERwin DM モデルでは、たとえば、属性を表すのに属性オブジェクトのインスタンスを使用します。属性の名前は、属性オブジェクトの Name プロパティに含まれています。

オブジェクト識別子

各オブジェクトには識別子が必要です。識別子はオブジェクト インスタンスを一意に識別する値です。内部的には、オブジェクト識別子の長さは 20 バイトです。オブジェクト識別子には 2 つのコンポーネントが含まれています。最初の 16 バイトは GUID（UUID とも呼びます）、最後の 4 バイトは 32 ビットの符号なし整数接尾辞です。

GUID には次のコンポーネントが含まれています。

- 32 ビットの符号なし整数 1 つ
- 16 ビットの符号なし整数 2 つ
- 8 ビットの符号なし整数 8 つ（符号なし文字で表されます）

これらのコンポーネントの合計は 128 ビット（16 バイト）になります。末尾には、追加で 32 ビットの符号なし整数（4 バイトの接尾辞）が付きます。したがって、オブジェクト識別子の長さは合計 160 ビット（20 バイト）です。

オブジェクト識別子を簡単に扱えるように、また COM オートメーションのデータタイプ制限のために、API では文字列を使用してオブジェクト識別子を表します。

次の表に、このドキュメントおよびインターフェイス定義で使用されるエイリアスを示します。

タイプ名	フォーマット	使用
SC_OBJID	{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}+接尾辞	オブジェクト識別子
SC_CLSID	{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}+接尾辞	クラス（オブジェクト、プロパティ タイプなど）識別子
SC_MODELTYPEID	{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}+接尾辞	モデル タイプ識別子
SC_CREATORID	{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}	作成者識別子

これらの識別子のうち GUID コンポーネントにゼロを含むものは、事前定義済みのオブジェクト識別子セットです。識別子の最後の 4 バイトにもゼロが含まれている場合、その識別子は NULL 識別子を表します。他のオフセット値は、今後の使用のために予約されています。

オブジェクト識別子とタイプ コード

SCD_SL_ M0 レイヤのオブジェクト インスタンスと、SCD_SL_ M1 レイヤのオブジェクト インスタンスとの間のリレーションシップについて検討してみましょう。SCD_SL_ M0 レイヤのインスタンスは、SCD_SL_ M1 レイヤのインスタンスによって記述されます。たとえば、SCD_SL_ M1 レイヤの 1 つのオブジェクトは、SCD_SL_ M0 レイヤのすべてのエンティティ インスタンスを記述します。

すべてのタイプ コードもオブジェクト識別子であるため、それらは同じフォーマットである必要があります。

プロパティ、プロパティ フラグ、および値ファセット

プロパティは、値と追加フラグの形式でデータを表します。

プロパティ値は 1 つの値を持つスカラか、あるいは複数値を持つ非スカラのいずれかになります。スカラおよび非スカラ プロパティ値の詳細については、「スカラおよび非スカラ プロパティ値」セクションを参照してください。

プロパティ値は、プロパティ タイプ（文字列や整数など）によって定義されます。プロパティ タイプの詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションを参照してください。

追加のプロパティ フラグには、2 つのタイプがあります。

プロパティ レベル フラグ

プロパティについての読み取り専用の情報を表します。次の例は、プロパティ レベル フラグで表すプロパティ インスタンスについての情報の一部です。

メタデータ情報

メタデータのプロパティがユーザー定義であるか、またはスカラ値を含むかを示します。

プロパティ状態の情報

プロパティが読み取り専用であるかどうかを示します。

データ ソース情報

データ ソースが計算済みであるかどうかを示します。

プロパティ値レベル フラグ

プロパティ値についての更新可能な情報を表します。

個々のプロパティ レベル フラグは、プロパティ フラグ値のビット フィールドで表されます。フラグは確認専用です。変更することはできません。特定のプロパティ フラグの詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションを参照してください。

値レベル フラグ、つまりファセットは、プロパティ値に関する追加データを表します。たとえば、プロパティ値が「固定」されており、継承によって変更されないという状態などです。

個々のファセットは数値 ID または名前によって識別され、3 つの状態（未設定、TRUE、または FALSE）から、いずれか 1 つの状態を取ります。

ファセットはプロパティ値の一部として扱われます。プロパティに新しい値を割り当てると、すべてのファセットが未設定の状態で配置されます。同様に、ある値を更新するか削除すると、すべてのファセットが未設定の状態になります。各プロパティのファセットの組み合わせは 1 つしかありません。、スカラ、または非スカラのいずれかです。非スカラ プロパティの個々の値を変更しても、プロパティ ファセットには影響ありません。

特定の値ファセットの詳細については、付録 A「API インターフェイスのリファレンス」の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

スカラおよび非スカラ プロパティ値

スカラ プロパティは、1 つの値で表されるプロパティです。複数値（同じタイプまたは異なるタイプ）を含むプロパティを非スカラ プロパティと呼びます。

プロパティのタイプを判別するには、プロパティ フラグを確認します。スカラ プロパティには SCD_MPF_SCALAR フラグがあります。

特定のプロパティ フラグの詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションを参照してください。

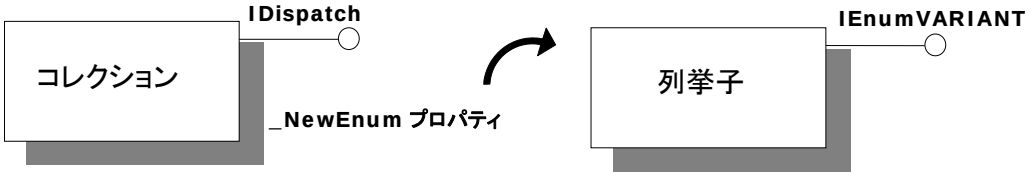
スカラ プロパティの値、または非スカラ プロパティのメンバにアクセスするには、ISCModelProperty インターフェイスの Value プロパティを使用します。

注: r7 では、異なるタイプから構成される非スカラ プロパティはサポートしません。非スカラ プロパティのメンバは、常に同じデータタイプになります。

スカラまたは非スカラ プロパティは、特別な NULL 値を持つことができます。NULL 値を持つプロパティには、SCD_MPF_NULL フラグ セットがあります。

コレクションとオートメーション

オートメーションに定義された IEnumVARIANT インターフェイスは、API クライアントがコレクションに対して処理を繰り返す標準的な方法を提供します。API のすべてのコレクション インターフェイスには、_NewEnum という名前の読み取り専用プロパティがあり、API クライアントはそのコレクションが繰り返しをサポートするかを判別できます。_NewEnum プロパティは IEnumVARIANT インターフェイスのポインタを返します。



IEnumVARIANT インターフェイスを使用すると、1 つのコレクションに含まれたすべての項目を通じて処理を繰り返すことができます。このインターフェイスは、コレクションの _NewEnum プロパティによって返される列挙子インターフェイスでサポートされます。

IEnumVARIANT インターフェイスは、次のメンバ関数を定義します。

Next

現在の要素から開始して、コレクションの 1 つ以上の要素を取得するように指定します。

Skip

コレクションの 1 つ以上の要素をスキップするように指定します。

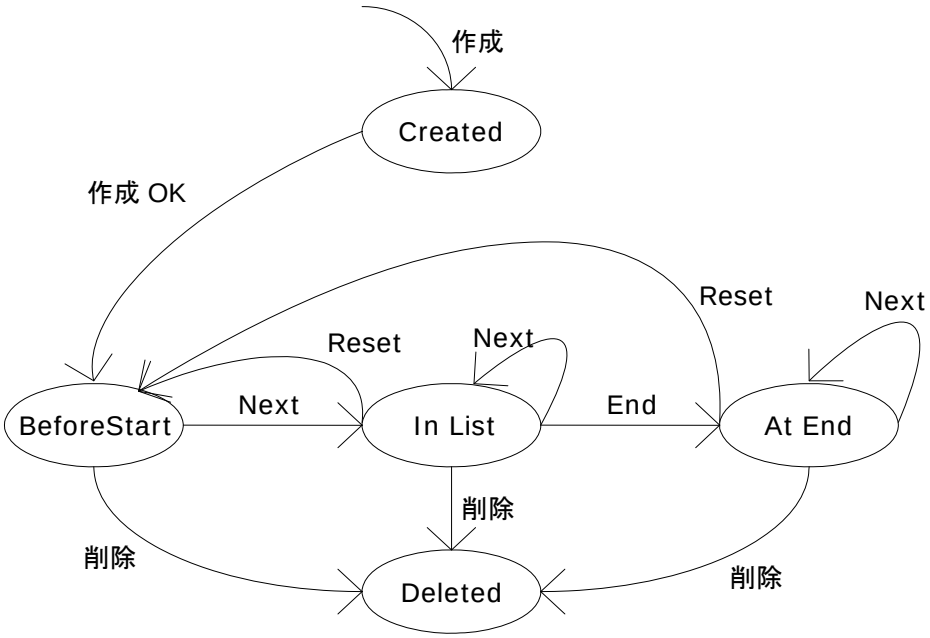
Reset

現在の要素をコレクションの最初の要素にリセットするように指定します。

Clone

列挙値の現在の状態をコピーして、Skip または Reset の使用後に現在の要素に戻ることができるように指定します。

IEnumVARIANT コレクションは、Rogue Wave Software, Inc. スタイルの“advance and return”繰り返し処理を実装しています。このため、処理のライフ サイクルは次のようになります。



IPtUCModelPropIter のライフサイクル

反復子が作成されると、Created 状態になり、次に BeforeStart 状態になります。advance に成功すると反復子は InList 状態になります。advance に失敗すると AtEnd 状態になります。Reset すると反復子は BeforeStart 状態に戻り、Delete すると Deleted 状態になります。

反復子がコレクションのメンバに対して配置される(すなわち、現在のメンバに関連付けられる)のは、InList 状態にある場合のみです。

コレクション オブジェクトの_NewEnumプロパティ

_NewEnum プロパティは、IEnumVARIANT インターフェイスを通じて繰り返し処理をサポートするかを識別します。_NewEnum プロパティには次の要件があります。

- 名前が_NewEnum である。
- 列挙子 IUnknown インターフェイスへのポインタを返す。
- プロパティのディスパッチ識別子が DISPID = DISPID_NEWENUM (-4)である。

デフォルト プロパティ

オートメーションのデフォルト プロパティは、明示的なプロパティまたはメソッド コールなしにオブジェクトが参照されるときにアクセスされるプロパティです。プロパティのディスパッチ識別子は DISPID_VALUE です。

その他のパラメータ

オートメーション クライアントの要件をサポートするために、その他のパラメータはすべて VARIANT で表されます。このため、インターフェイスの説明におけるパラメータ タイプは、VARIANT 構造体で予期されるタイプを記載したものにすぎません。

API のサンプル クライアント

2 つの Visual Basic 6.0 サンプル プロジェクトが API に付属して提供されています。ERwinSpy_Sample.vbp および ErwinSpy_Addin_Sample.vbp です。

インストールで[カスタム]セットアップ タイプを選択して、インストールするプログラム機能の選択画面で[ERwin API サンプル クライアント]を選択します。インストール後に、Sample Client サブディレクトリにある 2 つのサンプル Visual Basic 6.0 プロジェクトにアクセスできます。このパスは AllFusion ERwin Data Modeler r7\Samples\ERwin API\Sample Client です。

APIサンプル クライアントの使用

このセクションでは、API サンプル クライアントをスタンドアロン バージョンおよびアドイン コンポーネントとして使用する方法を説明します。

サンプル プログラムのスタンドアロン バージョンは ERwinSpy_Sample.vbp です。ERwinSpy_Sample.vbp をビルドして ERwinSpy_Sample.exe プログラムを作成できます。このプログラムは AllFusion ERwin DM モデルのデータ ブラウザです。メタモデル、モデル データ、およびモデル オブジェクトとそれらのプロパティのような内部データを確認するために使用できます。

ERwinSpy_Sample.exe を使用すると、[File]メニューの[Open]をクリックして、*.ERWIN ファイルを開くことができます。モデルを開くか、[Models]メニューから選択すると、モデルに含まれるモデル オブジェクトが左側のペインに表示されます。オブジェクトをダブルクリックして、モデル オブジェクトの階層構造(親と子)を確認できます。オブジェクトのプロパティを確認するには、プログラム画面の左右ペインの間にあるボタンをクリックします。

[Models]メニューには、現在開いているメモリ内モデルが表示されます。メニュー項目の上部を使用してモデル データにアクセスできます。メニュー項目の下部を使用して、モデルに関連したメタモデル情報にアクセスできます。

サンプル プログラムのアドイン バージョンは ErwinSpy_Addin_Sample.vbp です。ERwinSpy_Addin_Sample.vbp をビルドしてアドイン コンポーネントを作成できます。アドイン コンポーネントを実行するには、[ツール]メニューの[アドイン]からコンポーネントを選択します。ErwinSpy_Addin_Sample.vbp プロジェクトを使用してアドイン コンポーネントをビルドしたら、メニューに登録する必要があります。

アドイン コンポーネントの登録

ErwinSpy_Addin_Sample.vbp プロジェクトを使用してアドイン コンポーネントをビルドしたら、メニューに登録する必要があります。

アドイン コンポーネントを登録するには

1. [ツール]メニューの[アドイン]をポイントし、[カスタマイズ]をクリックします。

   [アドイン マネージャ]ダイアログ ボックスが開きます。

2. ツールバーの[メニュー項目の追加]  ボタンをクリックします。

   [新規メニュー項目の追加]ダイアログ ボックスが開きます。

3. [名前]に「アドイン サンプル」、[COM プログラム ID]に「ERwinSpy_AddIn_Sample.ErwinSpy」、[関数名]に「実行」と入力します。[メニュー タイプ]ボックスの[COM オブジェクト]をクリックして、[OK]をクリックします。

   [新規メニュー項目の追加]ダイアログ ボックスが閉じて、[アドイン マネージャ]ダイアログ ボックスに戻ります。

4. [保存]をクリックします。

   設定が保存され、[アドイン マネージャ]ダイアログ ボックスが閉じます。

この手順を完了すると、[ツール]メニューの[アドイン]以下に新しいメニュー項目が作成されます。メニュー項目の名前は、[新規メニュー項目の追加]ダイアログ ボックスの[名前]フィールドに入力したテキストと同じになります。この場合は「アドイン サンプル」です。この新しいメニュー項目を選択すると、アドイン サンプル コンポーネントを有効にできます。このアドインの[Models]メニューには、AllFusion ERwin DM で開いているモデルが表示されます。

ERwin Spy

ERwin Spy アプリケーションは、メタデータ情報を視覚化してモデル固有のメタデータを表示します。API の機能を使用したサンプル アプリケーションであり、モデル データがどのように保存されているかを理解するのに役立つ機能があります。ERwin Spy を使用して AllFusion ERwin DM メタモデルの内容を確認すると、モデル オブジェクト、プロパティ、および相互参照が複雑に関連する AllFusion ERwin DM モデルの詳細を理解するのに役立ちます。AllFusion ERwin DM のインストール時に、ERwin Spy ユーティリティをインストールするかどうかを選択できます。

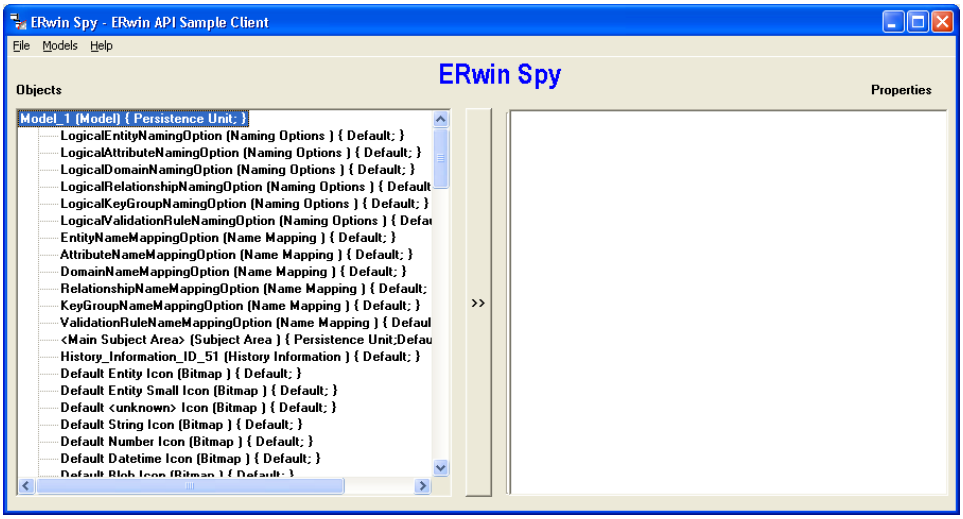
ERwin Spy フォルダで利用できるユーティリティには 2 つのバージョンがあります。スタンドアロン バージョンの ERwinSpy.7.0.exe と、アドイン バージョンの ERwinSpy_AddIn.7.0.dll です。

これら 2 つのバージョンの機能は同一ですが、アプリケーションの起動方法に違いがあります。スタンドアロン バージョンの動作には AllFusion ERwin DM が不要で、.erwin ファイルに保存されたモデルにアクセスできます。アドイン バージョンは AllFusion ERwin DM の[ツール]メニューから起動し、AllFusion ERwin DM で開いているモデルと.erwin ファイルに保存されたモデルにアクセスできます。

ERwin Spyアプリケーションの動作

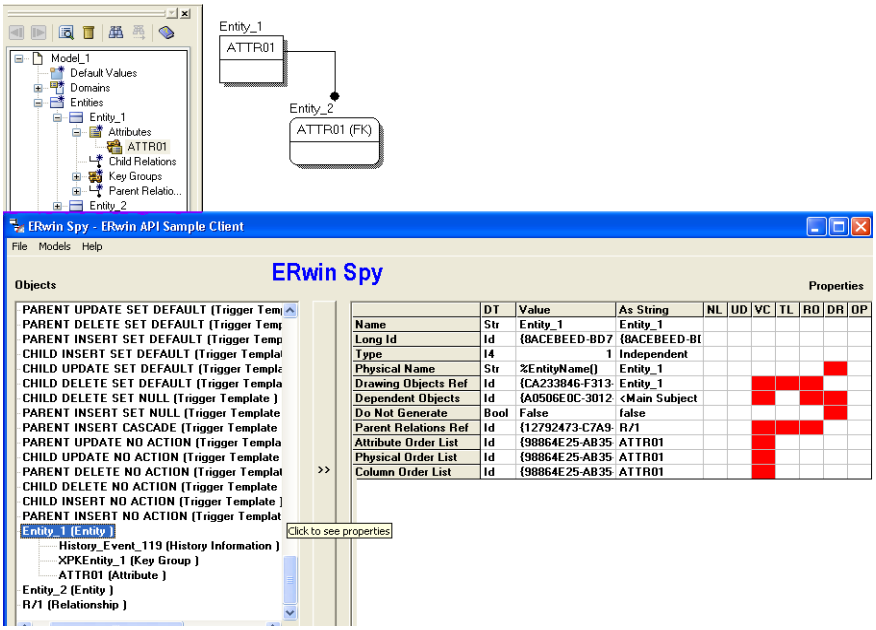
ERwin Spy でメタデータ情報を表示するには、次の手順に従います。

- AllFusion ERwin DM で空の論理/物理モデルを新規作成します。
- [ツール]メニューの[アドイン]をポイントし、[ERwin Spy]をクリックして起動します。
- ERwin Spy の[Models]メニューから一番上の項目を選択します。これは先に作成した空のモデルです。
- 左側ペインにあるモデル オブジェクトをダブル クリックして展開します。次のような画面が表示されます。



多数のオブジェクトが一覧表示されます。空のモデルであっても、AllFusion ERwin DM にデフォルトで存在するオブジェクトが表示されます。ドメイン、メイン サブジェクト エリア、トリガ テンプレートなどです。デフォルト オブジェクトには、モデル オブジェクト タイプの右側に { Default; }フラグが付きます。

ERwin Spy の右側ペインには、オブジェクトのプロパティが表示されます。特定のオブジェクトのプロパティを表示するには、オブジェクトを選択して画面中央のボタンをクリックします。選択したオブジェクトのプロパティが右側ペインに表示されます。次の図に、モデルに追加されたエンティティのプロパティを示します。



一番左のカラムはプロパティ名を示します。Name、Long ID、Type、Physical Name などのプロパティがあります。

2 番目のカラム DT は、プロパティのデータタイプを示します。たとえば、文字列は Str、数値は I4、ブール値は Bool、他のオブジェクトへの参照は Id です。

3 番目のカラム Value は、プロパティ値をネイティブ形式で示します。

4 番目のカラム As String は、文字列として再解釈されたプロパティ値を示します。このカラムの意味を理解するには、左側のカラムの Physical Name を確認します。Value カラムの値は%EntityName()であり、これはマクロです。As String カラムの値はマクロの展開形である Entity_1 です。

右側ペインの残りのカラムは、プロパティ フラグを示します。これらのカラムの意味を説明します。

次のカラム名はプロパティ フラグを表し、ERwin Spy の右側ペインに表示されます。

NL

NULL/値なしのプロパティであることを示します。**注:** このフラグは ERwin Spy ではオンになりません。

UD

ユーザー定義プロパティであることを示します。

VC

ベクトル プロパティであることを示します。

TL

AllFusion ERwin DM によって管理されるプロパティであり、API を使用して直接変更できないことを示します。

RO

読み取り専用プロパティであることを示します。

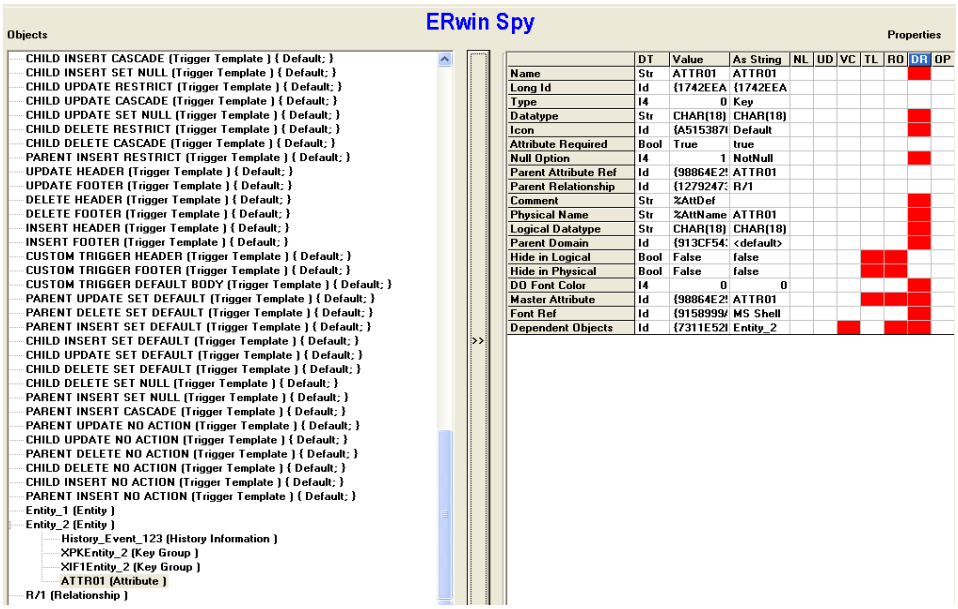
DR

（たとえば、親ドメインから）値が継承された導出プロパティであることを示します。

OP

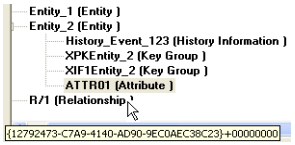
プロパティが任意であり削除可能であることを示します。

前の図では、主キー属性「ATTR01」が Entity_1 に追加され、依存型リレーションシップによって Entity_2 に移行されました。ERwin Spy で Entity_2 をダブルクリックして、「ATTR01」を選択します。画面中央のボタンをクリックすると、この属性のプロパティが右側ペインに表示されます。



Entity_2 の「ATTR01」は、外部キーの移行によって生成された属性なので、Parent Relationship Ref プロパティは、移行に使用されたリレーションシップ オブジェクトを示します。DT カラムの値 Id は、このプロパティが参照であることを示します。つまり、このプロパティの値はリレーションシップ オブジェクトのユニーク ID です。

As String カラムの名前を確認するか、ユニーク ID からオブジェクトを特定すると、ダイアグラム上の該当するリレーションシップ オブジェクトが見つかります。オブジェクト ID を確認するには、[File]メニューの[Options]をポイントし、[Show Ids]をクリックします。このオプションを有効にすると、左ペインのオブジェクト名の上にマウス カーソルを置いたときに、オブジェクトのユニーク ID がポップアップ ウィンドウに表示されます。



次に、Parent Relationship Ref プロパティと、Parent Attribute Ref および Master Attribute プロパティを比較します。Master Attribute プロパティは読み取り専用です。確認用にのみ表示され、API を使用して値を変更することはできません。モデルの作成時にモデル内のオブジェクトを展開して、AllFusion ERwin DM がプロパティをどのように使用して、モデル内の異なるリレーションシップを表しているかを確認することができます。

ERwin Spy ユーティリティを使用して、API で利用可能な AllFusion ERwin DM モデルのデータの詳細を確認することができます。AllFusion ERwin DM モデルで特定のデータがどのように表されているかを確認するには、先に説明した手順に従います。空のモデルから開始して、問題となる機能を表すのに必要な最小限のモデルを作成します。次に、ERwin Spy を使用して、表示されるデータの詳細を確認します。

第3章　API タスク

この章では、API を使用して基本的なタスクを実行する方法を説明します。各タスクの説明に加えて、必要なインターフェイスとメソッドの一覧を表に示します。ほとんどの場合、表に示すのはインターフェイスの一部のメソッドのみです。すべての API インターフェイスとメソッドの一覧については、付録 A「API インターフェイスのリファレンス」を参照してください。

本章には次のトピックがあります。

API 環境

API は COM DLL セットとしてパッケージ化されており、ユーザー プロセスとして動作します。EAL.dll は API 環境の起動を制御します。AllFusion ERwin DM をインストールすると、EAL.dll と他の API コンポーネントが、AllFusion ERwin Data Modeler r7 ディレクトリにコピーされ、API がシステム レジストリに登録されます。

開発環境で API を使用するには、EAL.dll ファイルのリソースとして組み込まれた API タイプライブラリを使用します。この操作は使用する言語に依存します。詳細については、ご使用の開発環境のマニュアルを参照してください。

API は 2 つのモードで動作します。スタンドアロン モードとアドイン モードです。

スタンドアロン モードでは、自身のプロセスをホストするクライアント アプリケーションが API を有効化および制御します。

アドイン モードでも、クライアント アプリケーションが API を有効化および制御しますが、クライアント アプリケーションは COM DLL として実装されます。すべてのクライアント アプリケーション DLL は、AllFusion ERwin DM が所有するプロセスの内部で実行されます。COM DLL をアドイン モードで使用するには、システム レジストリ、および AllFusion ERwin DM のアドイン マネージャに登録する必要があります。

2 つのモードでの API コンポーネントの動作は基本的に同じです。いくつかの例外については、このセクションで説明します。

API は COM インターフェイスのツリーとして実装されます。アプリケーションはトップレベルのインターフェイスをエクスポートし、クライアントはそこから必要に応じて低レベルのインターフェイスを取得します。

ISCApplication オブジェクトの作成

API のインターフェイス階層へのエントリ ポイントには、ISCApplication インターフェイスを使用します。ISCApplication インターフェイスから、永続ユニットおよびセッションへアクセスできます。API の他のインターフェイスを使用するには、はじめに ISCApplication のインスタンスを作成する必要があります。

例 1

次の例は、C++で ISCApplication オブジェクトを作成する方法を示します。

```
#import "EAL.dll" using namespace SCAPI;
ISCApplicationPtr scAppPtr;
HRESULT hr;
hr = scAppPtr.CreateInstance(__uuidof(SCAPI::Application));
```

次の例は、Visual Basic .NET で ISCApplication オブジェクトを作成する方法を示します。

```
Dim scApp As SCAPI.Application
scApp = New SCAPI.Application

// または、プログラム ID を使用する
Dim oApp As Object
oApp = CType(CreateObject("AllFusionERwin.SCAPI"), SCAPI.Application)
```

アプリケーション プロパティ

AllFusion ERwin DM アプリケーションについての情報を取得するには、次の表を使用します。

ISCApplicationインターフェイス

次の表に、ISCApplication インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
BSTR Name()	モデリング アプリケーションの名前	なし
BSTR Version()	モデリング アプリケーションのバージョン	なし
BSTR ApiVersion()	API のバージョン	なし

シグネチャ	説明	有効な引数
ISCApplicationEnvironment ApplicationEnvironment()	ランタイム環境の属性および利用可能な機能をレポートします（たとえばアドイン モード、ユーザー インターフェイスの表示設定など）。	なし
ISCPersistenceUnitCollection * PersistenceUnits()	アプリケーションに読み込まれたすべての永続ユニットのコレクションを返します。	なし
ISCSessionCollection * Sessions()	アプリケーション内に作成されたセッションのコレクションを返します。	なし

ISCApplicationEnvironment

次の表に、ISCApplicationEnvironment インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPropertyBag PropertyBag(VARIANT Category[省略可], VARIANT Name[省略可], VARIANT AsString[省略可])	Category と Name で指定された 1 つ以上のプロパティ値を持つプロパティ バッグを追加します。	Category: ■ Empty - すべてのカテゴリの全機能セットを返します。 ■ VT_BSTR – 指定されたカテゴリの機能を返します。 Name: ■ Empty – 選択されたカテゴリの全プロパティを返します。 ■ VT_BSTR – 指定された名前とカテゴリのプロパティを返します。 AsString: ■ Empty – プロパティ バッグのすべての値はネイティブ型になります。 ■ VT_BOOL – TRUE に設定すると、プロパティ バッグのすべての値は文字列型になります。

PropertyBag プロパティの Category パラメータ内の機能カテゴリは階層構造になっており、ドット（.）を使用して機能のサブセットを定義します。たとえば、Application カテゴリは、AllFusion ERwin DM のすべての機能を含むプロパティ バッグを追加しますが、Application.API は API に関連するサブセットのみを追加します。

Category パラメータが設定されていない場合、PropertyBag プロパティは利用可能なすべてのカテゴリの全機能を返します。

例 2

次の例は、C++で API を使用して Application Features を取得する方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
void IteratePersistenceUnits(ISCApplicationPtr & scAppPtr)
{
ISCPropertyBagPtr scBag;

// アプリケーション環境のすべてのプロパティを 1 回の呼び出しで取得する
scBag = scAppPtr ->GetApplicationEnvironment()->GetPropertyBag();
// カテゴリ名に空の文字列を使用して、カテゴリの配列を取得する
scBag = scAppPtr ->GetApplicationEnvironment()->GetPropertyBag("",
"Categories")

// Application Api カテゴリから API バージョン値を取得する
scBag = scAppPtr ->GetApplicationEnvironment()->GetPropertyBag
("Application.Api","Api Version")
}
```

次の例は、Visual Basic .NET で API を使用して Application Features を取得する方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
Public Sub GetApplicationFeatures(ByRef scApp As SCAPI.Application)
Dim scBag As SCAPI.PropertyBag
' アプリケーション環境のすべてのプロパティを 1 回の呼び出しで取得する
scBag = scApp.ApplicationEnvironment.PropertyBag
' 値を取得する
PrintPropertyBag(scBag)
' カテゴリ名に空の文字列を使用して、カテゴリの配列を取得する
scBag = scApp.ApplicationEnvironment.PropertyBag("", "Categories")
'  バッグからカテゴリー覧を取得する
Dim aCategories() As String
Dim CategoryName As Object
If IsArray(scBag.Value("Categories")) Then
' 配列を取得する
aCategories = scBag.Value("Categories")
If aCategories.Length > 0 Then
' カテゴリに基づいて値を取得する
For Each CategoryName In aCategories
' カテゴリの値を持つプロパティ バッグを取得する
scBag = scApp.ApplicationEnvironment.PropertyBag(CategoryName)
Console.WriteLine("   Values for the " + CategoryName + " category:")
' 値を取得する
PrintPropertyBag(scBag)
Next CategoryName
End If
End If
' Application Api カテゴリから API バージョン値を取得する
scBag = scApp.ApplicationEnvironment.PropertyBag("Application.Api", "Api
Version")
' 値を取得する
PrintPropertyBag(oBag)
End Sub
' プロパティ バッグの値を取得および表示する
Public Sub PrintPropertyBag(ByRef oBag As SCAPI.PropertyBag)
Dim Idx As Short
Dim nIdx1 As Short
If Not (oBag Is Nothing) Then
For Idx = 0 To oBag.Count - 1
If IsArray(oBag.Value(Idx)) Then
' 配列を取得する
If oBag.Value(Idx).Length > 0 Then
Console.WriteLine(Str(Idx) + ") " + oBag.Name(Idx) + " is an array: ")
For nIdx1 = 0 To UBound(oBag.Value(Idx))
Console.WriteLine("     " + oBag.Value(Idx)(nIdx1).ToString)
Next nIdx1
End If
Else
' 値が 1 つの場合
Console.WriteLine(Str(Idx) + ") " + oBag.Name(Idx) + " = " +
oBag.Value(Idx).ToString)
End If
Next Idx
```

```
End If
End Sub
```

モデルへのアクセス

API クライアントがモデル データにアクセスするには、利用可能な永続ユニットのプールを使用します。永続ユニットは、1 つのモデルに関するすべてのデータを記述する API の概念です。永続ユニットは、ファイルや AllFusion Model Manager モデルのような永続ストレージにアクセスして、保存することができます。クライアントが永続ユニットを操作するには、Persistence Units コレクションを使用します。
アプリケーションに永続ユニットが存在するかどうかは、アプリケーションのインスタンスが作成された時点で決定されます。
たとえば、スタンドアロン モードでは、起動時に永続ユニットは存在しません。コレクション内のユニットを収集するには、ユニット コレクション インターフェイスのメソッドを使用する必要があります。アドイン コンポーネント モードでは、クライアント コンポーネントが有効になった時点で AllFusion ERwin DM ユーザー インターフェイスで分かっているすべてのユニットがコレクションに含まれます。

スタンドアロンモードでは、クライアント プログラムが終了すると、メモリ内に存在する永続ユニットはすべて閉じられます。アドイン コンポーネント モードでは、クライアント プログラムが終了してもユニットは開いたままとなり、AllFusion ERwin DM ユーザー インターフェイスで利用できます。ただし、プログラムの終了前に永続ユニット コレクションから明示的に閉じられるか削除された場合を除きます。

注: AllFusion ERwin DM では、コレクションは 1 つのスナップショットです。コレクションには、コレクションが構築された時点（たとえば、ISCApplication インターフェイスの PersistenceUnits メソッドが呼び出された時点）で存在するユニットのみが含まれています。ただし、コレクションに追加されたユニットやコレクションから削除されたユニットは例外です。これらの変更は反映されます。すべての新規コレクションにも、これらの変更が反映されます。

APIをアドイン ツールとして使用する

API クライアントが、[ツール]メニューの[アドイン]から呼び出される DLL である場合、このクライアントは AllFusion ERwin DM の環境内で動作します。AllFusion ERwin DM で開いているすべてのモデルは、ISCApplication インターフェイスのインスタンスが作成される際に、PersistenceUnits プロパティに追加されます。

AllFusion ERwin DM で開いているすべてのモデルに処理を繰り返すには、ISCApplication インターフェイス、ISCPersistenceUnitCollection インターフェイス、および ISCPersistenceUnit インターフェイスを使用します。これらについては次に説明します。

ISCApplication インターフェイス

次の表に、ISCApplication インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPersistenceUnitCollection PersistenceUnits()	アプリケーションに読み込まれたすべての永続ユニットのコレクションを返します。	なし

ISCPersistenceUnitCollection インターフェイス

次の表に、ISCPersistenceUnitCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPersistenceUnit Item(VARIANT nIndex)	順序位置によって特定された PersistenceUnit コンポーネントのポインタを返します。	Index: ■ VT_UNKNOWN – セッションへのポインタ。セッションに関連した永続ユニットを取得します。 ■ VT_I4 – コレクション内のインデックス。コレクション インデックスの有効値は、0 から、サイズから 1 を引いた値までです。指定されたインデックスでコレクションの永続ユニットを取得します。 ■ VT_BSTR – アプリケーション レベルで一意な永続ユニットの識別子。
long Count()	コレクション内の永続ユニット数	なし

ISCPersistenceUnit インターフェイス

次の表に、ISCPersistenceUnit インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
BSTR Name()	永続ユニットの名前を返します。	なし
SC_MODELTYPEID ObjectId()	永続ユニットの識別子を返します。	なし
ISCPropertyBag PropertyBag(VARIANT List[省略可], VARIANT AsString[省略可])	永続ユニットのプロパティを持つプロパティ バッグを返します。	List: ■ VT_BSTR – セミコロンで区切られたプロパティ名の一覧。指定された一覧のユニット プロパティを持つプロパティ バッグを返します。 AsString: ■ VT_BOOL – TRUE に設定すると、すべての値を文字列型としたプロパティ バッグを返します。それ以外の場合、すべての値はネイティブ型になります。

シグネチャ	説明	有効な引数
VARIANT_BOOL HasSession()	ユニットに 1 つ以上の接続済みセッションがある場合、TRUE を返します。	なし
VARIANT_BOOL IsValid()	自分自身が有効な場合、TRUE を返します。	なし

永続ユニットのプロパティ バッグ メンバ

次の表に、既存の永続ユニットのプロパティ バッグ メンバであるプロパティ名とその説明をいくつか示します。

注: 利用可能なプロパティの一覧については、付録 A「API インターフェイスのリファレンス」を参照してください。

プロパティ名	タイプ	説明
Locator	BSTR	永続ユニットの場所(ファイル名など)を返します。一度も保存されていない新規モデルなど、永続的な場所を持たないモデルでは利用できません。
Disposition	BSTR	永続ユニットのディスポジション(読み取り専用など)を返します。
Persistence Unit Id	SC_MODELTYPEID	永続ユニットの識別子を取得します。
Model Type	Long	永続ユニットのタイプを取得します(たとえば、論理モデル、論理/物理モデル、物理モデルなど)。
Target Server Target Server Version Target Server Minor Version	Long	物理モデルと論理/物理モデルの対象データベースのプロパティを取得します。
Active Model	ブール値	永続ユニットが現在のモデルを表し、AllFusion ERwin DM ユーザー インターフェイスでアクティブである場合、TRUE となります。スタンドアロン モードの API では利用できません。
Hidden Model	ブール値	永続ユニット データを持つモデル ウィンドウが AllFusion ERwin DM ユーザー インターフェイスで非表示である場合、TRUE となります。スタンドアロン モードの API では利用できません。
Active Subject Area and Stored Display	SAFEARRAY(BSTR)	アクティブなサブジェクト エリアとストアド ディスプレイのモデル オブジェクト名をレポートします。これは、AllFusion ERwin DM の画面に表示されているサブジェクト エリアとストアド ディスプレイを指します。戻り値は、2 つの要素を持つセーフ配列です。最初の要素はアクティブなサブジェクト エリアの名前で、2 つめの要素はアクティブなストアド ディスプレイの名前です。

ISCPropertyBag インターフェイス

次の表に、ISCPropertyBag インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
long Count()	プロパティの数を返します。	なし
VARIANT Value(VARIANT Property)	バッグ内の指定されたプロパティを取得します。	Property: ■ VT_BSTR – プロパティ名。プロパティ バッグ内の指定された名前を持つプロパティの値です。 ■ VT_I4 – ゼロベースのプロパティ インデックス。プロパティ バッグ内の指定されたインデックスを持つプロパティの値です。
BSTR Name(long PropertyIdx)	指定されたインデックスを持つプロパティ名を取得します。インデックスの範囲は、0 から、サイズから 1 を引いた値までです。	なし

例 3

次の例は、C++で API をアドイン ツールとして使用して、開いているすべてのモデルに処理を繰り返す方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
void IteratePersistenceUnits(ISCApplicationPtr & scAppPtr)
{
ISCPersistenceUnitCollectionPtr scPUnitColPtr;
scPUnitColPtr = scAppPtr->GetPersistenceUnits();

ISCPersistenceUnitPtr scPUnit = 0;
long lCnt = scPUnitColPtr->GetCount();

for(long i = 0; i < lCnt; i++)
    {
scPUnit = scPUnitColPtr->GetItem(i);
CString csName = scPUnit->GetName();   // モデルの名前
ISCPropertyBagPtr scPropBag = scPUnit->GetPropertyBag("Locator;Active  Model");
long index = 0;
CComVariant vPathName = scPropBag->GetValue(ColeVariant(index));
//モデルの完全パス
index = 1;
CComVariant cActiveModel = scPropBag->GetValue(COleVariant(index)); // アクティブ モデルでは TRUE
        // …
    }
}
```

次の例は、Visual Basic .NET で API をアドイン ツールとして使用して、開いているすべてのモデルに処理を繰り返す方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
Public Sub IteratePersistenceUnits(ByRef scApp As SCAPI.Application)

Dim scPersistenceUnitCol as SCAPI.PersistenceUnits

Dim numUnits As Integer
Dim scPUnit As SCAPI.PersistenceUnit

scPersistenceUnitCol = scApp.PersistenceUnits

' 開いているユニットをカウント
numUnits = scPersistenceUnitCol.Count
If (numUnits > 0) Then
        For Each scPUnit In scPersistenceUnitCol
              Dim propBag As SCAPI.PropertyBag

              propBag = scPUnit.PropertyBag("Locator")
              Console.WriteLine( persUnit.Name )   ' モデルの名前
              Console.WriteLine( propBag.Value(0)) ' モデルの完全パス
               ' …
         Next
End If
End Sub
```

APIをスタンドアロンの実行ファイルとして使用する

API クライアントがスタンドアロンの実行ファイルの場合、クライアントは AllFusion ERwin DM 環境の外部で動作します。したがって、ISCApplication インターフェイスが作成されると、PersistenceUnits プロパティは空のコレクションになります。AllFusion ERwin DM が実行中で、開いているモデルがある場合でも、PersistenceUnits プロパティは空になります。スタンドアロンの API 環境は AllFusion ERwin DM 環境と無関係であるためです。有効な永続ユニットを取得するには、API クライアントで新規モデルを作成するか、または既存のモデルを開く必要があります。

モデルを作成する

API を使用して新規モデルを作成するには、最初に ISCPropertyBag の新規インスタンスを作成する必要があります。ISCPropertyBag インターフェイスは、新規モデルのプロパティを保持するプロパティ バッグです。次のプロパティを使用して新規モデルを作成します。

注: すべてのプロパティの一覧については、付録 A「API インターフェイスのリファレンス」を参照してください。

プロパティ名	タイプ	説明
Model Type	Long	永続ユニットのタイプを設定します。 ■ 1 – Logical（論理モデル。タイプを指定しない場合のデフォルトにもなります） ■ 2 – Physical（物理モデル） ■ 3 – Combined（論理/物理モデル）
Target Server Target Server Version Target Server Minor Version	Long	物理モデルと論理/物理モデルの対象データベースのプロパティを設定します。 詳細については、付録 A「API インターフェイスのリファレンス」を参照してください。

プロパティ バッグを作成して値を設定したら、永続ユニット コレクション内に新規永続ユニットを作成する必要があります。

ISCPersistenceUnitCollection インターフェイス

次の表に、ISCPersistenceUnitCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPersistenceUnit * Create(ISCPropertyBag * PropertyBag, VARIANT ObjectId [省略可])	新規ユニットを作成して、コレクションに登録します。	ObjectId: ■ Empty – API は新規永続ユニットに ID を割り当てます。 ■ VT_BSTR – API は新規永続ユニットに指定された ID を割り当てます。

ISCPropertyBag インターフェイス

次の表に、ISCPropertyBag インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Add(BSTR Name, VARIANT Value)	新規プロパティをバッグに追加します。	Value: すべての VARIANT が有効です。プロパティがバッグに追加された場合、TRUE を返します。そうでない場合は FALSE を返します。

例 4

次の例は、C++で新規永続ユニットを作成する方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
ISCPersistenceUnitPtr CreateNewModel(ISCApplicationPtr & scAppPtr)
{
ISCPersistenceUnitCollectionPtr scPUnitColPtr;
scPUnitColPtr = scAppPtr->GetPersistenceUnits();

ISCPropertyBagPtr propBag;
HRESULT hr =propBag.CreateInstance(__uuidof(SCAPI::PropertyBag));
if (FAILED(hr))
return;
propBag->Add("Name", “Test Model”);
propBag->Add("ModelType", “Logical”);
ISCPersistenceUnitPtr scPUnitPtr = scPUnitColPtr->Create(propBag,vtMissing);
return scPUnitPtr;
}
```

次の例は、Visual Basic .NET で新規永続ユニットを作成する方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
Public Function CreateNewModel(ByRef scApp As SCAPI.Application) As
SCAPI.PersistenceUnit
Dim scPersistenceUnitCol as SCAPI.PersistenceUnits
scPersistenceUnitCol = scApp.PersistenceUnits

Dim propBag As New SCAPI.PropertyBag

propBag.Add("Name", "Test Model")
propBag.Add("ModelType", 0)
CreateNewModel = scPersistenceUnitCol.Create(propBag)
End Function
```

既存モデルを開く

既存の AllFusion ERwin DM モデルを開くには、永続ユニットを永続ユニット コレクション（ISCPersistenceUnitCollection）に追加します。API クライアントがアドイン ツールの場合、API を通じてモデルを開くと、AllFusion ERwin DM ユーザー インターフェイス上にモデルが開きます。

ISCPersistenceUnitCollection インターフェイス

次の表に、ISCPersistenceUnitCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPersistenceUnit * Add(VARIANT Locator, VARIANT Disposition [省略可])	新規永続ユニットをユニット コレクションに追加します。	Locator: ■ VT_BSTR – AllFusion ERwin DM モデルへの完全パス。これは永続ユニットに読み込まれるモデルです。 Disposition: ■ VT_BSTR – 読み取り専用などのアクセス属性を指定します。 Locator パラメータの詳細と Disposition パラメータの形式については、付録 A「API インターフェイスのリファレンス」を参照してください。

例 5

次の例は、C++で既存のモデルを開く方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
ISCPersistenceUnitPtr OpenModel(ISCApplicationPtr & scAppPtr, CString &
csFullPath)
{
ISCPersistenceUnitCollectionPtr scPUnitColPtr;
scPUnitColPtr = scAppPtr->GetPersistenceUnits();
ISCPersistenceUnitPtr scPUnitPtr = scPUnitColPtr-
>Add(COleVariant(csFullPath));
return scPUnitPtr;
}
```

次の例は、Visual Basic .NET で既存のモデルを開く方法を示します。例 1 で作成された Application オブジェクトを使用します。

```
Public Function OpenModel(ByRef scApp As SCAPI.Application, _
fullModelPath As String) As SCAPI.PersistenceUnit
Dim scPersistenceUnitCol as SCAPI.PersistenceUnits
scPersistenceUnitCol = scApp.PersistenceUnits

OpenModel = scPersistenceUnitCol.Add(fullModelPath)
End Sub
```

セッションを開く

API を使用してモデル オブジェクトにアクセスするには、あらかじめ、モデルの ISCPersistenceUnit に対して ISCSession インスタンスを確立する必要があります。永続ユニットにセッションを開くには、新規 ISCSession を ISCSessionCollection に追加してから、新規セッションの ISCPersistenceUnit を開きます。

ISCSessionCollection インターフェイス

次の表に、ISCSessionCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCSession * Add()	新しい Session オブジェクトを作成し、コレクションに追加します。	なし

ISCSession インターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Open(IUnknown * Unit, VARIANT Level [省略可], VARIANT Flags [省略可])	Unit パラメータで特定された永続ユニットに自身をバインドします。	Unit: ■ 読み込まれた永続ユニットへのポインタ。永続ユニットをセッションにアタッチします。 Level: ■ Empty – データ レベル アクセス(SCD_SL_M0)がデフォルトになります。 ■ SCD_SL_M0 – データ レベル アクセス。 Flags: ■ Empty – SCD_SF_NONE がデフォルトになります。 ■ SCD_SF_NONE – アタッチされた永続ユニットに他のセッションがアクセスできることを指定します。 ■ SCD_SF_EXCLUSIVE – アタッチされた永続ユニットに他のセッションがアクセスできないことを指定します。

例 6

次の例は、C++でセッションを開く方法を示します。例 1 で作成された Application オブジェクトと、例 4 の CreateNewModel 関数を使用します。

```
ISCSessionPtr OpenSession(ISCApplicationPtr & scAppPtr)
{
ISCSessionCollectionPtr scSessionColPtr = scAppPtr->GetSessions();
ISCSessionPtr scSessionPtr = scSessionColPtr->Add();  // 新規セッションの追加
ISCPersistenceUnitPtr scPUnitPtr = CreateNewModel(scAppPtr); // 例 4 から

CComVariant varResult = scSessionPtr->Open(scPUnitPtr, (long) SCD_SL_M0); // ユニットを開く
if (varResult.vt == VT_BOOL && varResult.boolVal == FALSE)
return NULL;
return scSessionPtr;
}
```

次の例は、Visual Basic .NET でセッションを開く方法を示します。例 1 で作成された Application オブジェクトと、例 4 の CreateNewModel 関数を使用します。

```
Public Function OpenSession(ByRef scApp As SCAPI.Application) As SCAPI.Session
Dim scSessionCol As SCAPI.Sessions
Dim scPUnit As SCAPI.PersistenceUnit
scSessionCol = scApp.Sessions
OpenSession = scSessionCol.Add  ' 新規セッション

scPUnit = CreateNewModel(scApp)  ' 例 4 から
scSession.Open(scPUnit, SCD_SL_M0)  ' 永続ユニットを開く
End Sub
```

モデル セットにアクセスする

1 つの永続ユニットに含まれているデータは、リンクされたモデル セットのグループです。モデル セットはツリー型の階層構造であり、階層のトップには 1 つのモデル セットが配置されます。

永続ユニット内のトップ モデル セットには、ほとんどのモデリング データが含まれています。AllFusion ERwin DM の API では、EMX という略語を使用してトップ モデル セットを表します。

EMX モデル セットにはセカンダリ モデル セットがあります。これは EM2 という略語で表され、ユーザー オプションとユーザー インターフェイス設定が含まれています。

ISCSession インターフェイスを使用すると、ISCSession::Open の呼び出しで ISCPersistenceUnit インターフェイスへのポインタを指定して、トップ モデル セットを簡単に開くことができます。

永続ユニットを構成するすべてのモデル セットに対して、繰り返し実行することができます。繰り返しの間、ISCModelSet インターフェイスへのポインタを使用して、特定のモデル セットとのセッションを開くことができます。具体的には、そのポインタを、永続ユニットではなく ISCSession::Open 呼び出しの第 1 パラメータとして送信します。

ISCPersistenceUnit インターフェイスの ModelSet プロパティは、永続ユニットのモデル セットに対して処理を繰り返すための開始ポイントになります。ISCModelSet の OwnedModelSets プロパティを使用すると、永続ユニットにある次のレベルのモデル セットに処理を繰り返すことができます。

ISCPersistenceUnit インターフェイス

次の表に、ISCPersistenceUnit インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelSet * ModelSet()	永続ユニットのトップ モデル セットのポインタを返します。	なし

ISCModelSet インターフェイス

次の表に、ISCModelSet インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelSetCollection * OwnedModelSets()	直接所有するモデル セットのコレクションを提供します。	なし

ISCModelSetCollection インターフェイス

次の表に、ISCModelSetCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelSet * Item(VARIANT nIndex)	ModelSet コンポーネントのポインタを返します。	nIndex: ■ VT_I4 – モデル セット コレクション内のモデルセットのインデックス。ゼロベース インデックスです。 ■ VT_BSTR – モデル セット識別子。 ■ VT_BSTR – モデル セットに関連付けられたメタデータのクラス識別子。 ■ VT_BSTR – モデル セットに関連付けられたメタデータのクラス名。 メタデータのクラス識別子と名前についての詳細は、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCSession インターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Open(IUnknown * ModelSet, VARIANT Level [省略可], VARIANT Flags [省略可])	ModelSet パラメータで特定されたモデル セットに自身をバインドします。	ModelSet: ■ 読み込まれた永続ユニット内のモデル セットへのポインタ。モデル セットをセッションにアタッチします。 Level: ■ Empty – データ レベル アクセス(SCD_SL_M0)がデフォルトになります。 ■ SCD_SL_M0 – データ レベル アクセス。 Flags: ■ Empty – SCD_SF_NONE がデフォルトになります。 ■ SCD_SF_NONE – アタッチされた永続ユニットに他のセッションがアクセスできることを指定します。 ■ SCD_SF_EXCLUSIVE – アタッチされた永続ユニットに他のセッションがアクセスできないことを指定します。

例 7

次の例は、C++で永続ユニットの EM2 モデルとのセッションを開く方法を示します。例 1 で作成された Application オブジェクトと、例 4 の CreateNewModel 関数を使用します。

```
void OpenEM2(ISCApplicationPtr & scAppPtr)
{
ISCSessionCollectionPtr scSessionColPtr = scAppPtr->GetSessions();
ISCPersistenceUnitPtr scPUnitPtr = CreateNewModel(scAppPtr); // 例 4 から

ISCModelSetPtr scEMXModelSetPtr = scPUnitPtr->ModelSet(); // トップ モデル セット
を収集
ISCModelSetPtr scEM2ModelSetPtr = scEMXModelSetPtr-
>GetOwnedModelSets()->GetItem(COleVariant("EM2"));
if (scEM2ModelPtr != NULL)
    {
ISCSessionPtr scSessionPtr = scSessionColPtr->Add();  // 新規セッションの追加
CComVariant varResult = scSessionPtr->Open(scEM2ModelSetPtr);
if (varResult.vt == VT_BOOL && varResult.boolVal == FALSE)
return;

     // …
}
```

次の例は、Visual Basic .NET で永続ユニットの EM2 モデルとのセッションを開く方法を示します。例 1 で作成された Application オブジェクトと、例 4 の CreateNewModel 関数を使用します。

```
Public Sub OpenEM2(ByRef scApp As SCAPI.Application )

Dim scSession As SCAPI.Session
Dim scEMXModelSet As SCAPI.ModelSet
Dim scEM2ModelSet As SCAPI.ModelSet
Dim scPUnit As SCAPI.PersistenceUnit

scSessionCol = scApp.Sessions
scPUnit = CreateNewModel(scApp)  ' 例 4 から

' EMX タイプのトップ モデル セットにアクセス
scEMXModelSet = persUnit.ModelSet
' 所有する EM2 モデル セットにクラス名でアクセス
scEM2ModelSet = scEMXModelSet.OwnedModelSets("EM2")
Console.WriteLine(vbTab + " Access EM2 Model Set by class name" +
scEM2ModelSet.Name + _" Id " + scEM2ModelSet.ModelSetId)
Console.WriteLine(vbTab + vbTab + " Class Name " + scEM2ModelSet.ClassName + _"
Class id " + scEM2ModelSet.ClassId)
scSession = scSessionCol.Add  ' 新規セッション
scSession.Open(scEM2ModelSet, SCD_SL_M0)  ' EM2 をセッションに接続する
    '…
End Sub
```

モデル オブジェクトへのアクセス

モデル オブジェクトにアクセスするには、アクティブな ISCSession インスタンスの ModelObjects プロパティを使用します。ModelObjects プロパティは、セッションの永続ユニットに関連付けられたすべてのモデル オブジェクトのコレクションです。ModelObjects プロパティは、ISCModelObjectCollection のインスタンスです。ISCModelObjectCollection のインスタンスの繰り返し処理は、深さ優先の方式で実行され、ISCModelObject のインスタンスを返します。

次のセクションで、モデル オブジェクトのアクセスに使用するインターフェイスを説明します。

ISCSessionインターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelObjectCollection * ModelObjects()	セッションの ModelObjectコレクションを作成します。	なし

ISCModelObjectCollectionインターフェイス

次の表に、ISCModelObjectCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
long Count()	コレクション内のオブジェクト数	なし
IUnknown _NewEnum()	コレクション列挙子オブジェクトのインスタンスを作成します。	なし

ISCModelObjectインターフェイス

次の表に、ISCModelObject インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
BSTR ClassName()	現在のオブジェクトのクラス名を返します。	なし
SC_OBJID ObjectId()	現在のオブジェクトを一意に識別します。	なし
BSTR Name()	現在のオブジェクトの名前または文字列識別子を返します。	なし
SC_CLSID ClassId()	現在のオブジェクトのクラス識別子を返します。	なし

シグネチャ	説明	有効な引数
ISCModelObject * Context()	オブジェクトのコンテキスト(親)を返します。	なし

例 8

次の例は、C++でモデル オブジェクトにアクセスする方法を示します。例 1 で作成された Application オブジェクトと、例 6 の OpenSession 関数を使用します。

```
void IterateObjects(ISCApplicationPtr & scAppPtr)
{
ISCSessionPtr scSessionPtr = OpenSession( scAppPtr );    // 例 6 から
// セッション Ptr が開いていることを確認
if(!scSessionPtr->IsOpen())
    {
AfxMessageBox("Session Not Opened");
return;
    }
ISCModelObjectCollectionPtr scModelObjColPtr = scSessionPtr >GetModelObjects();
IUnknownPtr _NewEnum = NULL;
IEnumVARIANT* ObjCollection;

_NewEnum = scModelObjColPtr ->Get_NewEnum();
if (_NewEnum != NULL)
    {
HRESULT hr = _NewEnum->QueryInterface(IID_IEnumVARIANT, (LPVOID*)
&ObjCollection);
if (!FAILED(hr))
       {
COleVariant xObject;
while (S_OK == ObjCollection->Next(1,&xObject,NULL))
          {
ISCModelObjectPtr pxItem = (V_DISPATCH (&xObject));
// xObject の ISCModelObject は既に AddRefed であった
// それをスマート ポインタにアタッチする必要がある
xObject.Clear();
// 項目を処理
CString csName = (LPSTR) pxItem->GetName();
CString csID = (LPSTR) pxItem->GetObjectId();
CString csType = (LPSTR) pxItem->GetClassName();
              // …
          }
if (ObjCollection)
ObjCollection->Release();
    }
}
```

次の例は、Visual Basic .NET でモデル オブジェクトにアクセスする方法を示します。例 1 で作成された Application オブジェクトと、例 6 の OpenSession 関数を使用します。

```
Public Sub IterateObjects(ByRef scApp As SCAPI.Application)
Dim scSession As SCAPI.scSession
Dim scModelObjects As SCAPI.ModelObjects
Dim scObj As SCAPI.ModelObject

scSession = OpenSession( scApp )        ' 例 6 から
' セッションが開いていることを確認
If scSession.IsOpen() Then
scModelObjects = scSession.ModelObjects

For Each scObj In scModelObjects
Console.WriteLine( scObj.Name )
Console.WriteLine( scObj.ObjectId )
Dubug.WriteLine( scObj.ClassName )
Next
End If
End Sub
```

特定のオブジェクトにアクセスする

ISCModelObjectCollection インスタンスのモデル オブジェクトに直接アクセスするには、インターフェイスの Item メソッドを使用します。

ISCModelObjectCollection インターフェイス

次の表に、ISCModelObjectCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelObject * Item(VARIANT nIndex, VARIANT Class [省略可])	nIndex パラメータによって特定された Model Object コンポーネントの IUnknown ポインタを返します。	nIndex: ■ VT_UNKNOWN – ISCModelObject インターフェイスへのポインタ。コレクションから指定されたオブジェクトを返します。 ■ VT_BSTR – オブジェクト ID。コレクションから指定された識別子を持つオブジェクトを返します。 ■ VT_BSTR – オブジェクト名。オブジェクト名を使用する場合、Class パラメータも使用する必要があります。コレクションから指定された名前と Class タイプを持つオブジェクトを返します。 Class: ■ Empty – コレクションから nIndex で指定されたオブジェクトを返します。 ■ VT_BSTR – クラス名。nIndex パラメータがオブジェクト名である場合に使用する必要があります。指定された名前と Class を持つオブジェクトを返します。有効なオブジェクト クラス名については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。 ■ VT_BSTR – オブジェクト タイプの Class ID。nIndex パラメータがオブジェクト名である場合に使用する必要があります。指定された名前と Class 識別子を持つオブジェクトを返します。有効なオブジェクト クラス識別子については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

例 9

次の例は、C++で特定のオブジェクトにアクセスする方法を示します。例 6 の Session オブジェクトを使用します。

```
void GetObject(ISCSessionPtr & scSessionPtr, CString & csID)
{
ISCModelObjectCollectionPtr scModelObjColPtr = scSessionPtr->GetModelObjects();
ISCModelObjectPtr scObjPtr = scModelObjColPtr->GetItem(COleVariant(csID));
    // …
}
```

次の例は、Visual Basic .NET で特定のオブジェクトにアクセスする方法を示します。例 6 の Session オブジェクトを使用します。

```
Public Sub GetObject(ByRef scSession As SCAPI.Session, ByRef objID As String)
Dim scObjCol as SCAPI.ModelObjects
Dim scObj as SCAPI.ModelObject

scObjCol = scSession.ModelObjects
scObj = scObjCol.Item(objID)  ' 指定されたオブジェクト ID を持つオブジェクトを取得
End Sub
```

オブジェクト コレクションのフィルタ

コレクションのサブセットを作成するには、ISCModelObjectCollection::Collect メソッドを使用します。Collect メソッドは、メソッドのパラメータで指定されたフィルタ条件に基づき、Model Objects コレクション コンポーネントの新規インスタンスを作成します。フィルタ条件は省略可能です。条件はいくつでも組み合わせて使用することができます。

ISCModelObjectCollection インターフェイス

次の表に、ISCModelObjectCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelObjectCollection * Collect(VARIANT Root, VARIANT ClassId [省略可], VARIANT Depth [省略可], VARIANT MustBeOn [省略可], VARIANT MustBeOff [省略可])	自身のサブコレクションである Model Objects コレクションを作成します。 このメソッドは、コレクションが空の場合でも有効なコレクションを作成します。	Root: ■ VT_UNKNOWN – ルート オブジェクトの ISCModelObject ポインタ。指定されたオブジェクトの子を返します。 ■ VT_BSTR – ルート オブジェクトのオブジェクト ID。指定されたオブジェクト識別子を持つオブジェクトの子を返します。 ClassId: ■ VT_ARRAY|VT_BSTR – クラス ID の SAFEARRAY。指定されたオブジェクト クラス識別子を持つルートの子を返します。有効なオブジェクト識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。 ■ VT_ARRAY|VT_BSTR – クラス名の SAFEARRAY。指定されたオブジェクト クラス名を持つルートの子を返します。有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。 ■ VT_BSTR – クラス ID。指定されたオブジェクト クラス識別子を持つルートの子を返します。 ■ VT_BSTR – セミコロンで区切られたクラス ID の一覧。指定されたクラス識別子を持つルートの子を返します。 ■ VT_BSTR – クラス名。指定されたクラス名を持つルートの子を返します。 ■ VT_BSTR – セミコロンで区切られたクラス名の一覧。指定されたクラス名を持つルートの子を返します。 ■ Empty – クラス タイプに関係なく、すべての子を返します。

シグネチャ	説明	有効な引数
		Depth: ■ VT_I4 – 最大の深さ。指定された深さまでのルートの子を返します。深さが-1 の場合、深さの制限がないことを表します。 ■ Empty – ルートのすべての子を返します（深さの制限なし）。 MustBeOn: ■ VT_I4 – 指定されたオブジェクト フラグ セットを持つルートの子を返します。SC_ModelObjectFlags の詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションを参照してください。 ■ Empty – SCD_MOF_DONT_CARE がデフォルトになります。 MustBeOff: ■ VT_I4 – 指定されたオブジェクト フラグ セットを持たないルートの子を返します。 ■ Empty – SCD_MOF_DONT_CARE がデフォルトになります。

さまざまなフィルタを使用したコード例を次に示します。

例 10

次の例は、C++で Object Type をフィルタする方法を示します。例 6 の Session オブジェクトを使用して、rootObj オブジェクトが所有する csType タイプのオブジェクト コレクションを作成します。

```
void FilterObjects(ISCSessionPtr scSessionPtr, ISCModelObjectPtr & rootObj,
CString & csType)
{
ISCModelObjectCollectionPtr scModelObjectsPtr;
scModelObjectsPtr =
scSessionPtr->GetModelObjects()->Collect(rootObj->GetObjectId(),
COleVariant(csType));
    // …
}
```

次の例は、Visual Basic .NET で Object Type をフィルタする方法を示します。例 6 の Session オブジェクトを使用して、rootObj オブジェクトが所有する csType タイプのオブジェクト コレクションを作成します。

```
Public Sub FilterObjects(ByRef scSession As SCAPI.Session, _
ByRef rootObj As SCAPI.ModelObject, ByRef objType as String)

Dim scModelObjects As SCAPI.ModelObjects
scModelObjects = scSession.ModelObject.Collect(rootObj, objType)
' scModelObjects は objType タイプのオブジェクトのみを含む

End Sub
```

例 11

次の例は、C++で Depth をフィルタする方法を示します。

```
void FilterObjects(ISCSessionPtr scSessionPtr, ISCModelObjectPtr & rootObj,
CString & csType, long depth)
{
ISCModelObjectCollectionPtr scModelObjectsPtr;
scModelObjectsPtr = scSessionPtr->GetModelObject()->
Collect(rootObj->GetObjectId(), COleVariant(csType),depth);
    // …
}
```

次の例は、Visual Basic .NET で Depth をフィルタする方法を示します。

```
Public Sub FilterObjects(ByRef scSession As SCAPI.Session, _
ByRef rootObj As SCAPI.ModelObject, ByRef classID As String, depth As Integer)

Dim scModelObjects As SCAPI.ModelObjects
scModelObjects = scSession.ModelObjects.Collect(rootObj, classID, depth)

End Sub
```

例 12

次の例は、C++で MustBeOn/MustBeOff をフィルタする方法を示します。例 6 の Session オブジェクトを使用します。

```
void FilterObjects(ISCSessionPtr scSessionPtr, ISCModelObjectPtr & rootObj, long
depth)
{
ISCModelObjectCollectionPtr scModelObjectsPtr;
scModelObjectsPtr = scSessionPtr->GetModelObjects()->
Collect(rootObj->GetObjectId(), vtMissing, depth, SCD_MOF_USER_DEFINED);
      // …
}
```

次の例は、Visual Basic .NET で MustBeOn/MustBeOff をフィルタする方法を示します。例 6 の Session オブジェクトを使用します。

```
Public Sub FilterObjects(ByRef scSession As SCAPI.Session, _
ByRef rootObj As SCAPI.ModelObject, depth As Integer)

Dim scModelObjects As SCAPI.ModelObjects
scModelObjects = scSession.ModelObjects.Collect(rootObj, , depth,
SCD_MOF_USER_DEFINED)

End Sub
```

オブジェクト プロパティへのアクセス

オブジェクト プロパティにアクセスするには、ISCModelObject の Properties プロパティを使用します。Properties プロパティは、ISCModelPropertyCollection のインスタンスです。ISCModelPropertyCollection には ISCModelProperty のインスタンスが含まれています。

プロパティの繰り返し

このセクションでは、プロパティの繰り返しに関連するインターフェイスを説明します。

ISCModelObject インターフェイス

次の表に、ISCModelObject インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelPropertyCollection * Properties()	利用可能なすべてのプロパティのプロパティ コレクションを返します。	なし

ISCModelPropertyCollection インターフェイス

次の表に、ISCModelPropertyCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
Long Count()	コレクション内のプロパティ数	なし
IUnknown _NewEnum()	コレクション列挙子オブジェクトのインスタンスを作成します。	なし

ISCModelProperty インターフェイス

次の表に、ISCModelProperty インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
BSTR ClassName()	プロパティのクラス名を返します。	なし
SC_CLSID ClassId()	プロパティのクラス識別子を返します。	なし
Long Count()	プロパティ内の値の数が含まれます。	なし
BSTR FormatAsString()	プロパティ値を文字列型に変換します。	なし

例 13

次の例は、C++でプロパティの繰り返し処理を行う方法を示します。例 9 の Model Object オブジェクトを使用します。

```
void IterateObjectProperties(ISCModelObjectPtr & scObjPtr)
{
ISCModelPropertyCollectionPtr propColPtr = scObjPtr->GetProperties();

// コレクションを通じて繰り返す
IUnknownPtr _NewEnum = NULL;
IEnumVARIANT* propCollection;

_NewEnum = propColPtr->Get_NewEnum();
if (_NewEnum != NULL)
    {
HRESULT hr = _NewEnum->QueryInterface(IID_IEnumVARIANT, (LPVOID*)
&propCollection);
if (!FAILED(hr))
        {
COleVariant xObject;
while (S_OK == propCollection->Next(1,&xObject,NULL))
           {
ISCModelPropertyPtr scObjPropPtr = (V_DISPATCH (&xObject));
xObject.Clear();
if (scObjPropPtr.GetInterfacePtr())
              {
CString csPropName = (LPSTR) scObjPropPtr->GetClassName();
CString csPropVal= (LPSTR) scObjPropPtr->FormatAsString();
                 // …
              }
}  // プロパティの繰り返し
        }
if (propCollection)
propCollection->Release();
    }
}
```

次の例は、Visual Basic .NET でプロパティの繰り返し処理を行う方法を示します。例 9 の Model Object オブジェクトを使用します。

```
Public Sub IterateObjectProperties(ByRef scObj As SCAPI.ModelObject)

Dim scObjProperties As SCAPI.ModelProperties
Dim scObjProp As SCAPI.ModelProperty
scObjProperties = scObj.Properties
For Each scObjProp In scObjProperties
Debug.WriteLine( scObjProp.ClassName )
Console.WriteLine( scObjProp.Name )
Next

End Sub
```

ISCModelPropertyインターフェイス

次の表に、ISCModelProperty インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
long Count()	プロパティ内の値の数が含まれます。 スカラ プロパティでは、プロパティ内の値の数は常に 1 つです。 要素がない非スカラ プロパティを持つことも可能です。この場合、プロパティ内の値の数はゼロになります。	なし
SC_ModelPropertyFlags Flags()	プロパティのフラグを返します。 SC_ModelPropertyFlags の詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションを参照してください。	なし

シグネチャ	説明	有効な引数
VARIANT Value(VARIANT ValueId [省略可], VARIANT ValueType [省略可])	指定されたプロパティ値を、要求された形式で取得します。プロパティのデータタイプの詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションの「SC_ValueTypes」を参照してください。	ValueId: ■ Empty – スカラ プロパティのみで有効です。 ■ VT_I4 – 同タイプの配列内のゼロベース インデックス。このインデックスで指定されたメンバの値を返します。 ValueType: ■ Empty – 戻り値にネイティブのデータタイプを指定します。 ■ SCVT_DEFAULT – 戻り値にネイティブのデータタイプを指定します。 ■ SCVT_BSTR – 戻り値を文字列に変換します。

例 14

次の例は、C++でスカラ プロパティ値にアクセスする方法を示します。例 13 の Model Property オブジェクトを使用します。

```
void GetScalarProperty(ISCModelPropertyPtr & scObjPropPtr)
{
if (scObjPropPtr->GetCount() <= 1)
    {
_bstr_t bstrPropVal= scObjPropPtr->FormatAsString();
       // …
    }
}
```

次の例は、Visual Basic .NET でスカラ プロパティ値にアクセスする方法を示します。例 13 の Model Property オブジェクトを使用します。

```
Public Sub GetPropertyElement(ByRef scObjProp As SCAPI.ModelProperty)

If (scObjProp.Flags And SCAPI.SC_ModelPropertyFlags.SCD_MPF_NULL) Then
Console.WriteLine( "The value is Null" )
Else
If (scObjProp.Flags And SCAPI.SC_ModelPropertyFlags.SCD_MPF_SCALAR) Then
Console.WriteLine( scObjProp.Value.ToString() )
Else
For j = 0 To scObjProp.Count-1
           Console.WriteLine( scObjProp.Value(j).ToString() )
Next
End If
End If

End Sub
```

非スカラ プロパティ値に処理を繰り返す

複数値（同じタイプまたは異なるタイプ）を含むプロパティを非スカラ プロパティと呼びます。非スカラ プロパティの個々の値にアクセスするには、ISCModelProperty インターフェイスの PropertyValues メンバを使用します。PropertyValues メンバは、ISCPropertyValueCollection のインスタンスです。ISCPropertyValueCollection の各メンバは、ISCPropertyValue のインスタンスです。ISCPropertyValue インターフェイスの ValueId メンバは、非スカラ プロパティの個々のプロパティ値を識別します。プロパティ タイプが構造体の場合、ValueId はゼロベース インデックスか、または非スカラ プロパティ値メンバの名前になります。

ISCModelProperty インターフェイス

次の表に、ISCModelProperty インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPropertyValueCollection * PropertyValues()	プロパティの値を返します。	なし

ISCPropertyValueCollection インターフェイス

次の表に、ISCPropertyValueCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
long Count()	コレクション内の値の数。	なし
IUnknown _NewEnum()	コレクション列挙子オブジェクトのインスタンスを作成します。	なし

ISCPropertyValue インターフェイス

次の表に、ISCPropertyValue インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT ValueId(VARIANT ValueType [省略可])	非スカラ プロパティの値を一意に識別します。	ValueType: ■ SCVT_I2 – プロパティが非スカラの場合、プロパティ インデックスの値を返します。 ■ SCVT_I4 – プロパティが非スカラの場合、プロパティ インデックスの値を返します。

シグネチャ	説明	有効な引数
		■ SCVT_BSTR – 利用可能な場合、非スカラ プロパティ メンバの名前、,またはメンバのインデックスを返します。 ■ SCVT_DEFAULT – プロパティが非スカラの場合、プロパティ インデックスの値を返します。 ■ Empty – SCVT_DEFAULT がデフォルトになります。
SC_CLSID PropertyClassId()	現在のプロパティのクラス識別子を返します。	なし
BSTR PropertyClassName()	現在のプロパティのクラス名を返します。	なし
VARIANT Value(VARIANT ValueType [省略可])	現在の値を指定された値タイプに変換します。 値のデータタイプの詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションの「SC_ValueTypes」を参照してください。	ValueType: ■ SCVT_DEFAULT – ネイティブ形式の値を指定します。 ■ SCVT_BSTR – プロパティ値の文字列表記です。 ■ Target Type – 変換の対象タイプを指定します。 ■ Empty – SCVT_DEFAULT がデフォルトになります。
SC_ValueTypes ValueType()	値のデフォルト タイプの識別子を返します。	なし
SC_ValueTypes ValueIdType()	値の識別子のデフォルト タイプの識別子を返します。	なし
SC_ValueTypes * GetSupportedValueTypes()	サポートする値タイプの一覧をグループ化し、SAFEARRAY として返します。	なし
SC_ValueTypes * GetSupportedValueIdTypes()	現在の値の識別子でサポートする値タイプの一覧をグループ化し、SAFEARRAY として返します。	なし

例 15

次の例は、C++で非スカラ プロパティ値にアクセスする方法を示します。例 13 の Model Property オブジェクトを使用します。

```
void IterateNonScalarProperties(ISCModelPropertyPtr & scObjPropPtr)
{
if (scObjPropPtr->GetCount() > 1)
   {
ISCPropertyValueCollectionPtr propVals = scObjPropPtr->GetPropertyValues();
long numVals = propVals->GetCount();
for (long i = 0; i < numVals; i++)
     {
ISCPropertyValuePtr propValPtr = propVals->GetItem(COleVariant(i));
VARIANT valType;
V_VT(&valType) = VT_I4;
V_I4(&valType) = SCVT_BSTR;
bstr_t bstrPropVal = propValPtr->GetValue(valType);

         // …
     }
   }
}
```

次の例は、Visual Basic .NET で非スカラ プロパティ値にアクセスする方法を示します。例 13 の Model Property オブジェクトを使用します。

```
Public Sub IterateNonScalarProperties(ByRef scObjProp As SCAPI.ModelProperty)
Dim scPropValue as SCAPI.PropertyValue

If (scObjProp.Count > 1) Then
For Each scPropValue In scObjProp.PropertyValues
        If (scPropValue.ValueIdType = SCVT_BSTR) Then
           Console.WriteLine( scPropValue.ValueId(SCVT_BSTR),": ", _
scPropValue.Value.ToString())
        Else
Console.WriteLine (scPropValue.Value.ToString())
        End If
Next
End If
End Sub
```

特定のプロパティにアクセスする

非スカラ プロパティの個々の値に直接アクセスするには、ISCPropertyValueCollection の Item メソッドを使用します。

ISCPropertyValueCollection インターフェイス

次の表に、ISCPropertyValueCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPropertyValue * Item(VARIANT ValueId)	プロパティ値コレクションから 1 つの値を返します。	ValueId: ■ VT_I4 – 非スカラ プロパティ メンバのインデックス。 ■ VT_BSTR – 非スカラ プロパティ メンバの名前。

注: AllFusion ERwin DM r7 では、非スカラ プロパティ メンバの名前付けをサポートしていません。

例 16

次の例は、C++で特定のプロパティにアクセスする方法を示します。例 9 の Model Object オブジェクトを使用します。

```
// この関数は、指定された名前のプロパティから指定されたインデックスを持つ特定の
// 値を取得します。
ISCPropertyValuePtr GetPropValue(ISCModelObjectPtr & scObjPtr, CString & csName,
int index)
{
ISCModelPropertyCollectionPtr propColPtr = scObjPtr->GetProperties();
ISCModelPropertyPtr scObjPropPtr = propColPtr->GetItem(COleVariant(csName));
ISCPropertyValueCollectionPtr propVals = scObjPropPtr->GetPropertyValues();
return propVals->GetItem(COleVariant(index));
}
```

次の例は、Visual Basic .NET で特定のプロパティにアクセスする方法を示します。例 9 の Model Object オブジェクトを使用します。

```
' この関数は、指定された名前のプロパティから指定されたインデックスを持つ特定の
' 値を取得します。
Public Function GetPropValue(ByRef scObj As SCAPI.ModelObject, ByRef propName As
String, _index As Integer) As SCAPI.PropertyValue
Dim scProp as SCAPI.ModelProperty
Set scProp = scObj.Properties.Item(propName)
Set GetPropValue = scProp.PropertyValues.Item(index)

End Function
```

プロパティのフィルタ

ISCModelPropertyCollection のインスタンスのサブセットを作成するには、ISCModelObject の CollectProperties メソッドを使用します。CollectProperties メソッドは、メソッドのパラメータで指定されたフィルタ条件に基づき、ISCModelPropertyCollection の新規インスタンスを作成します。プロパティ コレクションをフィルタして、特定のクラスのプロパティや、指定されたフラグ セットを持つ/持たないプロパティを取得することができます。フィルタ条件は省略可能です。条件はいくつでも組み合わせて使用することができます。プロパティ クラスで使用される識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。特定のプロパティ フラグの詳細については、付録 A「API インターフェイスのリファレンス」の「列挙体」セクションを参照してください。

ISCModelObject インターフェイス

次の表に、ISCModelObject インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCModelPropertyCollection * CollectProperties(VARIANT ClassIds [省略可], VARIANT MustBeOn [省略可], VARIANT MustBeOff [省略可])	必要なタイプのプロパティ コレクションを返します。	ClassIds: ■ Empty – オブジェクトのすべてのプロパティを返します。 ■ VT_ARRAY|VT_BSTR – プロパティ クラス ID の SAFEARRAY。指定されたプロパティ クラス識別子を持つプロパティを返します。 ■ VT_ARRAY|VT_BSTR – プロパティ名の SAFEARRAY。指定されたクラス名を持つプロパティを返します。 ■ VT_BSTR – プロパティ クラスの ID。指定されたプロパティ クラス識別子を持つプロパティを返します。 ■ VT_BSTR – プロパティ名。指定されたクラス名を持つプロパティを返します。 ■ VT_BSTR – セミコロンで区切られたプロパティ クラス ID の一覧。指定されたプロパティ クラス識別子を持つプロパティを返します。 ■ VT_BSTR – セミコロンで区切られたプロパティ名の一覧。指定されたクラス名を持つプロパティを返します。 MustBeOn: ■ Empty – SCD_MPF_DONT_CARE がデフォルトになり、すべてのプロパティを返します。 ■ VT_I4 – オンにする必要がある SC_ModelObjectFlags フラグ。指定されたフラグ セットを持つプロパティを返します。 MustBeOff: ■ Empty – SCD_MPF_NULL がデフォルトになり、すべてのプロパティを返します。 ■ VT_I4 – オフにする必要がある SC_ModelObjectFlags フラグ。指定されたフラグ セットを持たないプロパティを返します。

注: フィルタ条件を設定すると、データ取得の有効性に影響を与えることがあります。たとえば、MustBeOn フィルタを SCD_MPF_DERIVED に設定すると、計算済みと導出済みプロパティのみを含むコレクションが作成されます。計算済みと導出済みプロパティの評価をリクエストすると、コレクションで処理を繰り返す間、パフォーマンスが低下します。ただし、MustBeOff フィルタを同じ値である SCD_MPF_DERIVED に設定し、計算済みおよび導出済みのプロパティを除外すると、パフォーマンスは改善されます。

例 17

次の例は、C++でプロパティをフィルタする方法を示します。例 9 の Model Object オブジェクトを使用します。

```
void GetProperties(ISCModelObjectPtr & scObjPtr)
{

ISCModelPropertyCollectionPtr propColPtr;

propColPtr = scObjPtr->GetProperties();    // フィルタなし

VARIANT valFlags;
V_VT(&valFlags) = VT_I4;
V_I4(&valFlags) = SCD_MPF_SCALAR;

propColPtr = scObjPtr->CollectProperties(vtMissing, valFlags, vtMissing); // スカラ プロパティのみ
propColPtr = scObjPtr->CollectProperties(vtMissing, vtMissing, valType);  // 非スカラ プロパティのみ
}
```

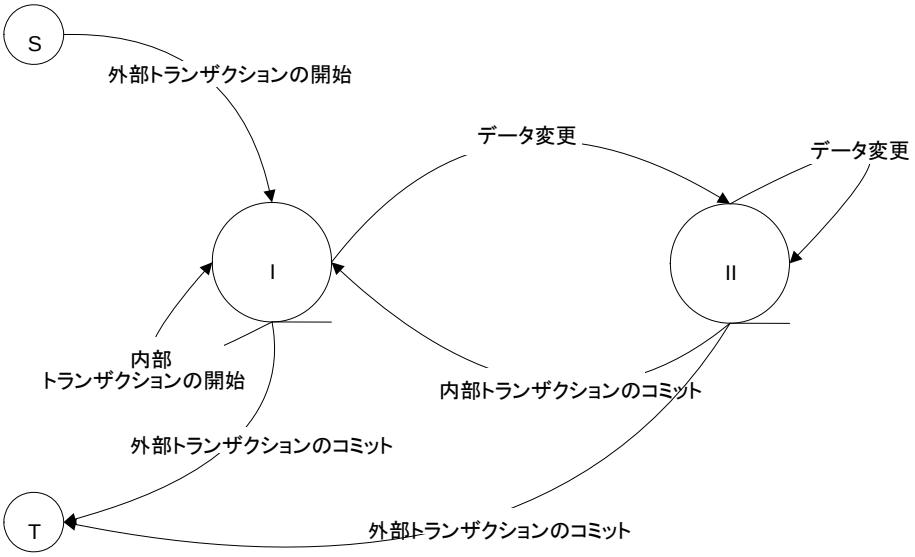
次の例は、Visual Basic .NET でプロパティをフィルタする方法を示します。例 9 の Model Object オブジェクトを使用します。

```
Public Sub( ByRef scObj As SCAPI.ModelObject )
Dim scObjProperties As SCAPI.ModelProperties

scObjProperties = scObj.Properties   ' フィルタなし

scObjProperties = scObj.CollectProperties(, SCD_MPF_SCALAR)  ' スカラ プロパティのみ

scObjProperties = scObj.CollectProperties(, , SCD_MPF_SCALAR)  ' 非スカラ プロパティのみ
End Sub
```

セッション トランザクションを使用してモデルを変更する

モデルに変更を加えるには、セッション トランザクションを使用する必要があります。変更を加える前に、BeginTransaction()または BeginNamedTransaction()を呼び出す必要があります。すべての変更が完了したら、CommitTransaction()を呼び出す必要があります。

注: r7 では、ネストしたトランザクションおよびロールバックがサポートされます（若干の制限事項があります）。次の状態図に、制限事項を示します。



外部トランザクションの開始後、API はダイアグラムの状態 I となります。新しくネストしたトランザクションを開くか、または外部トランザクションを閉じることができます。トランザクションを開く/閉じる以外の操作、たとえばオブジェクト/プロパティを作成したり変更すると、API は状態 II に移行します。状態 II では、さらに変更を続行することができます。ただし、トランザクションを新しくネストすることは許可されません。現在のトランザクションがコミットされるか、またはロールバックされるまで、API は状態 II に留まります。

ネストしたトランザクションを使用すると、変更フローを適切に制御することができます。次に例を示します。

トランザクションのコミット

登録された変更リストを直ちに実行します。これによって、外部トランザクションを閉じることなく、ネストした小さなトランザクションは複雑な変更の個々のステップを反映することができ、コミット済みのトランザクションはすぐに次のステップで使用することができます。

ロールバック

すべてのネストしたトランザクションの結果を取り消します。外部トランザクションのロールバック前にコミット済みである場合にも適用されます。

トランザクションの開始

モデルを変更するには、最初に BeginTransaction()または BeginNamedTransaction()を呼び出す必要があります。

ISCSession インターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT BeginTransaction()	セッションにトランザクションを開きます。トランザクションの識別子を返します。	なし
VARIANT BeginNamedTransaction(BSTR Name, VARIANT PropertyBag [省略可])	指定された名前のセッションにトランザクションを開きます。トランザクションの識別子を返します。	Name: 新規トランザクションの名前を指定します。 PropertyBag: トランザクションに対する任意のパラメータ コレクション。

例 18

次の例は、C++で BeginTransaction を使用してモデルを変更する方法を示します。例 6 の Session オブジェクトを使用します。

```
void OpenSession(ISCSessionPtr & scSessionPtr )
{
variant_t transactionId;   // セッションのトランザクション ID

VariantInit(&transactionId);
transactionId = scSessionPtr->BeginTransaction();

    // …
}
```

次の例は、Visual Basic .NET で BeginTransaction を使用してモデルを変更する方法を示します。例 6 の Session オブジェクトを使用します。

```
Public Sub OpenSession( ByRef scSession As SCAPI.Session )
Dim m_scTransactionId As Variant

scTransactionId = scSession.BeginNamedTransaction("My Transaction")
End Sub
```

トランザクションのコミット

CommitTransaction()を使用して、メモリ内モデルの変更をコミットします。

注: コミットが適用されるのは、API が実行中のメモリ内モデルのみです。変更を保存するには、ISCPersistenceUnit::Save()関数を使用して、モデルを明示的に保存する必要があります。

ISCSession インターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL CommitTransaction(VARIANT TransactionId)	指定されたトランザクションをコミットします。	なし

例 19

次の例は、C++で CommitTransaction を使用してモデルを変更する方法を示します。例 6 の Session オブジェクトを使用します。

```
void Transaction(ISCSessionPtr & scSessionPtr )
{
variant_t transactionId;   // セッションのトランザクション ID

VariantInit(&transactionId);
transactionId = scSessionPtr->BeginTransaction();

// ここでモデルに変更を加えます ….

scSessionPtr->CommitTransaction(transactionId);
}
```

次の例は、Visual Basic .NET で CommitTransaction を使用してモデルを変更する方法を示します。例 6 の Session オブジェクトを使用します。

```
Public Sub Transaction(ByRef scSession As SCAPI.Session )
Dim scTransactionId As Variant

scTransactionId = scSession.BeginTransaction

' ここでモデルに変更を加えます …

scSession.CommitTransaction( scTransactionId )
End Sub
```

オブジェクトの作成

新規オブジェクトを作成するには、最初に新規オブジェクトの親の ISCModelObject インスタンスを取得します。新規オブジェクトの親から、ISCModelObjectCollection のインスタンスにある子オブジェクトを取得します。次に、新規オブジェクトを子オブジェクトのコレクションに追加する必要があります。有効なオブジェクト クラス名とオブジェクト クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCModelObjectCollectionインターフェイス

次の表に、ISCModelObjectCollection インターフェイスの情報を示します。新規モデル オブジェクトの作成に使用します。

シグネチャ	説明	有効な引数
ISCModelObjectCollection * Collect(VARIANT Root, VARIANT ClassId [省略可], VARIANT Depth [省略可], VARIANT MustBeOn [省略可], VARIANT MustBeOff[省略可])	自身のサブコレクションである Model Objects コレクションを作成します。	Root: ■ VT_UNKNOWN – ルート オブジェクトの ISCModelObject ポインタ。指定されたオブジェクトの子を返します。 ■ VT_BSTR – ルート オブジェクトの ID。指定されたオブジェクト識別子を持つオブジェクトの子を返します。 ClassId: ■ Empty – オブジェクトの子を取得するときは不要です。 Depth: ■ VT_I4 – オブジェクト直下の子を取得するときは深さ 1 に設定します。 MustBeOn: ■ Empty – オブジェクトの子を取得するときは不要です。 MustBeOff: ■ Empty – オブジェクトの子を取得するときは不要です。
ISCModelObject * Add(VARIANT Class, VARIANT ObjectId [省略可])	Class タイプのオブジェクトをモデルに追加します。	Class: ■ VT_BSTR – クラス名。指定されたクラス名のオブジェクトを作成します。 ■ VT_BSTR – オブジェクト タイプの Class ID。指定された識別子を持つクラスのオブジェクトを作成します。 ObjectId: ■ Empty – API は新規オブジェクトにオブジェクト識別子を割り当てます。 ■ VT_BSTR – 新規オブジェクトの ID。API は指定されたオブジェクト識別子を新規オブジェクトに割り当てます。

例 20

次の例は、C++でオブジェクトを作成する方法を示します。例 6 の Session オブジェクトを使用します。

```
// 注: この関数を呼び出す前に、ISCSession::BeginTransaction()を呼び出す必要
// があります
// この関数から戻るときに、ISCSession::CommitTransaction()を呼び出す必要
// があります
void CreateObject(ISCSessionPtr & scSessionPtr, CString & csType,
ISCModelObjectPtr & parentObj)
{
variant_t transactionId;   // セッションのトランザクション ID
VariantInit(&transactionId);
transactionId = scSessionPtr->BeginTransaction();
ISCModelObjectCollectionPtr childObjColPtr =
scSessionPtr->GetModelObject()->Collect(parentObj->GetObjectId(),vtMissing,(lo
ng)1);  // 子オブジェクト
// を取得
// 子オブジェクトをコレクションに追加
ISCModelObjectPtr childObjPtr = childObjColPtr->Add(COleVariant(csType));
    // …
scSessionPtr->CommitTransaction(transactionId);
}
```

次の例は、Visual Basic .NET でオブジェクトを作成する方法を示します。例 6 の Session オブジェクトを使用します。

```
Public Sub AddNewObject(ByRef scSession As SCAPI.Session, _
ByRef parentObj As SCAPI.ModelObject, type As String)
Dim scObj as SCAPI.ModelObject
Dim scChildObjCol As SCAPI.ModelObjects
Dim transactionID as Variant

transactionID = scSession.BeginTransaction
scChildObjCol = scSession.ModelObjects.Collect(parentObj, , 1)   ' 子オブジェク
トのコレクション
scObj = scChildObjCol.Add(type)    ' 新規オブジェクトを子オブジェクトのコレクションに
追加

scSession.CommitTransaction( transactionID )
End Sub
```

プロパティ値の設定

モデル オブジェクトのプロパティ値を設定するには、ISCModelProperty インターフェイスのインスタンスの Value メンバを使用します。

スカラ プロパティ値を設定する

スカラ プロパティ値の設定に使用できる有効な VARIANT タイプは、プロパティのタイプに依存します。詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCModelProperty インターフェイス

次の表に、ISCModelProperty インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
void Value(VARIANT ValueId [省略可], VARIANT ValueType [省略可], VARIANT Val)	指定された値を持つプロパティ値を設定します。	ValueId: ■ Empty – スカラ プロパティの設定時には使用されません。 ValueType: ■ Empty – 未使用。 Val: ■ プロパティ タイプに依存します。有効な値の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

例 21

次の例は、C++でスカラ プロパティ値を設定する方法を示します。例 9 の Model Object オブジェクトを使用し、トランザクションが開いていると仮定します。

```
// 注: この関数を呼び出す前に、ISCSession::BeginTransaction()を呼び出す必要
// があります
// この関数から戻るときに、ISCSession::CommitTransaction()を呼び出す必要
// があります
void SetNameProperty(ISCModelObjectPtr & scObjPtr, CString & csName)
{
ISCModelPropertyCollectionPtr propColPtr = scObjPtr->GetProperties();
CString csPropName = "Name";
ISCModelPropertyPtr nameProp = propColPtr >  GetItem(COleVariant(csPropName));
if (nameProp != NULL)
nameProp->PutValue(vtMissing, (long) SCVT_BSTR, csName);
}
```

次の例は、Visual Basic .NET でスカラ プロパティ値を設定する方法を示します。例 9 の Model Object オブジェクトを使用し、トランザクションが開いていると仮定します。

```
' 注: この関数を呼び出す前に、ISCSession::BeginTransaction()を呼び出す必要があります
' この関数から戻るときに、ISCSession::CommitTransaction()を呼び出す必要があります
Public Sub SetScalarPropValue(ByRef scObj As SCAPI.ModelObject, ByRef val As
Variant)
Dim modelProp As SCAPI.ModelProperty
modelProp = scObj.Properties(“Name”)
modelProp.Value = val
End Sub
```

非スカラ プロパティ値を設定する

非スカラ プロパティ値を設定するには、設定したい値を特定する必要があります。これには、ValueId パラメータを使用します。プロパティが構造体の場合、ValueId はプロパティ値コレクションのゼロベース インデックスか、またはメンバの名前になります。

注: AllFusion ERwin DM r7 では、非スカラ プロパティ メンバの名前付けをサポートしていません。

ISCModelProperty インターフェイス

次の表に、ISCModelProperty インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
void Value(VARIANT ValueId [省略可], VARIANT ValueType [省略可], VARIANT Val)	指定された値を持つプロパティ値を設定します。	ValueId: ■ VT_I4 – 指定された値を設定する非スカラ プロパティのインデックス。 ■ VT_BSTR – 指定された値を設定する非スカラ プロパティのメンバの名前。 ValueType: ■ Empty – 未使用。 Val: ■ プロパティ タイプに依存します。有効な値については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

例 22

次の例は、C++で非スカラ プロパティ値を設定する方法を示します。例 9 の Model Object オブジェクトを使用し、トランザクションが開いていると仮定します。

```
// 注: この関数を呼び出す前に、ISCSession::BeginTransaction()を呼び出す必要
// があります
// この関数から戻るときに ISCSession::CommitTransaction()を呼び出す必要
// があります
void SetNameProperty(ISCModelObjectPtr & scObjPtr, CString & csValue)
{
ISCModelPropertyCollectionPtr propColPtr = scObjPtr->GetProperties();
CString csPropName = "Non-Scalar";
ISCModelPropertyPtr nameProp = propColPtr >  GetItem(COleVariant(csPropName));
if (nameProp != NULL)
// 最初の要素を設定
nameProp->PutValue(COleVariant(0L), (long) SCVT_BSTR, csValue);
}
```

次の例は、Visual Basic .NET で非スカラ プロパティ値を設定する方法を示します。例 9 の Model Object オブジェクトを使用し、トランザクションが開いていると仮定します。

```
' 注: この関数を呼び出す前に、ISCSession::BeginTransaction()を呼び出す必要があります
' この関数から戻るときに、ISCSession::CommitTransaction()を呼び出す必要があります
Public Sub SetScalarPropValue(ByRef scObj As SCAPI.ModelObject, ByRef val As
Variant)
Dim modelProp As SCAPI.ModelProperty
modelProp = scObj.Properties("Name")
Dim index As Long
Index = 0     ' インデックスをゼロに設定
modelProp.Value(index) = val   ' インデックスを使用して非スカラ プロパティにアクセス
End Sub
```

オブジェクトの削除

オブジェクトを削除するには、ISCModelObjectCollection のインスタンスから、オブジェクトの ISCModelObject インターフェイス インスタンスを削除します。削除するオブジェクトを特定するには、インターフェイスへのポインタ、またはオブジェクト識別子を使用します。

ISCModelObjectCollectionインターフェイス

次の表に、ISCModelObjectCollection インターフェイスの情報を示します。モデル オブジェクトの削除に使用します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Remove(VARIANT Object)	指定されたモデル オブジェクトをモデルから削除します。	Object: ■ VT_UNKNOWN – 削除するオブジェクトへの ISCModelObject * ポインタ。指定されたオブジェクトを削除します。 ■ VT_BSTR – オブジェクトの ID。指定されたオブジェクト識別子を持つオブジェクトを削除します。

例 23

次の例は、C++でオブジェクトを削除する方法を示します。モデル オブジェクト コレクションが存在しており、トランザクションが開いていると仮定します。

```
CString csID;   // 削除するオブジェクトの ID
// …
CComVariant bRetVal = scObjColPtr->Remove(COleVariant(csID));
```

次の例は、Visual Basic .NET でオブジェクトを削除する方法を示します。モデル オブジェクト コレクションが存在しており、トランザクションが開いていると仮定します。

```
bRetVal = scObjCol.Remove(objID)
```

プロパティとプロパティ値の削除

プロパティを削除するには、ISCModelPropertyCollection インターフェイスのインスタンスからプロパティを削除します。プロパティが非スカラの場合、個々のプロパティ値を削除するには、ISCModelProperty インターフェイスの RemoveValue メソッドを使用します。有効なプロパティ名とプロパティ ID の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

次のセクションで、モデル プロパティとモデル プロパティ値の削除に使用するインターフェイスを説明します。

ISCModelPropertyCollectionインターフェイス

次の表に、ISCModelPropertyCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Remove(VARIANT ClassId)	バインドされたオブジェクトから指定されたプロパティを削除します。	ClassId: ■ VT_UNKNOWN – 削除するオブジェクトへの ISCModelProperty ポインタ。指定されたプロパティを削除します。 ■ VT_BSTR – プロパティ名。指定されたクラス名を持つプロパティを削除します。 ■ VT_BSTR – プロパティの ID。指定されたクラス識別子を持つプロパティを削除します。

ISCModelPropertyインターフェイス

次の表に、ISCModelProperty インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL RemoveValue(VARIANT ValueId [省略可])	指定された値をプロパティから削除します。	ValueId: ■ Empty – スカラ プロパティでのみ有効です。 ■ VT_I4 – 非スカラ プロパティのインデックス。非スカラ プロパティから、指定されたインデックスを持つ値を削除します。 ■ VT_BSTR – 非スカラ プロパティのプロパティ メンバの名前。指定された名前を持つ非スカラ プロパティ メンバの値を削除します。
VARIANT_BOOL RemoveAllValues()	プロパティからすべての値を削除します。	なし

例 24

次の例は、C++でスカラ プロパティを削除する方法を示します。モデル オブジェクトが存在しており、トランザクションが開いていると仮定します。

```
CString propName("Some Property Name");
// …
CComVariant bRetVal = scObjPtr->GetProperties()->Remove(COleVariant(propName));
```

次の例は、Visual Basic .NET でスカラ プロパティを削除する方法を示します。モデル オブジェクトが存在しており、トランザクションが開いていると仮定します。

```
Dim propName As String
propName = "Some Property Name"
bRetVal = scObj.Properties.Remove(propName)
```

非スカラ プロパティ値を削除する

非スカラ プロパティからすべての値を削除するには、Remove メソッドを使用して ISCModelPropertyCollection からプロパティ自体を削除します。非スカラ プロパティから特定の値を削除するには、ISCModelProperty インターフェイスの RemoveValue メソッドを使用します。非スカラ プロパティ値にアクセスするときと同様に、プロパティ値の特定には ValueId パラメータを使用します。プロパティ タイプが構造体の場合、ValueId は値のゼロベース インデックスか、またはメンバの名前になります。

注: AllFusion ERwin DM r7 では、非スカラ プロパティ メンバの名前付けをサポートしていません。

例 25

次の例は、C++で非スカラ プロパティ値を削除する方法を示します。モデル オブジェクトが存在しており、トランザクションが開いていると仮定します。

```
ISCModelPropertyCollectionPtr propColPtr = scObjPtr->GetProperties();
CString csPropName = "Some Property Name";
ISCModelPropertyPtr scPropPtr = propColPtr->GetItem(COleVariant(csPropName));
long index;   // 非スカラ プロパティ メンバのインデックス
index = 0;    // 最初の要素を設定
// …
bRetVal = scPropPtr->RemoveValue(index);   // プロパティから 1 つの値を削除
```

次の例は、Visual Basic .NET で非スカラ プロパティ値を削除する方法を示します。モデル オブジェクトが存在しており、トランザクションが開いていると仮定します。

```
Dim scProp As SCAPI.ModelProperty
scProp = scObj.Properties("Some Property Name")
bRetVal = scProp.RemoveValue(index)  ' プロパティから 1 つの値を削除
```

モデルの保存

AllFusion ERwin DM モデルに変更を加えた場合、変更を保持するには永続ユニットを保存する必要があります。

ISCPersistenceUnitインターフェイス

次の表に、ISCPersistenceUnit インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Save(VARIANT Locator [省略可], VARIANT Disposition [省略可]	モデル データを外部ストレージに保存します。	Locator: ■ VT_BSTR – モデルを保存する場所の完全パス。永続ユニット データ ソースの新しい保存場所です。ファイルまたは AllFusion Model Manager 項目の場所を示す文字列で表し、ストレージへのアクセスに必要な属性も含まれます。 ■ Empty – 永続ユニットの元の場所を使用します。 Disposition: 読み取り専用など、アクセス属性の変更を指定します。

例 26

次の例は、C++でモデルを保存する方法を示します。例 5 の Persistence Unit オブジェクトを使用します。

```
void Save( ISCPersistenceUnitPtr & scPUnitPtr )
{
ISCPropertyBagPtr propBag = scPUnitPtr->GetPropertyBag ("Locator");
long index = 0;
_bstr_t bstrFileName = propBag->GetValue(COleVariant(index));

// bstrFileName を新しい場所に変更する
scPUnitPtr->Save(bstrFileName);
}
```

次の例は、Visual Basic .NET でモデルを保存する方法を示します。例 5 の Persistence Unit オブジェクトを使用します。

```
Public Sum Save( scPUnit As SCAPI.PersistenceUnit )
Dim propBag as SCAPI.PropertyBag
propBag = scUnit.PropertyBag(“Locator”)
Dim sFileName As String
sFileName = propBag.Value(“Locator”)
sFileName = sFileName + “.bak”
scPUnit.Save(sFileName )
End Sub
```

メタモデル情報へのアクセス

API を使用して、AllFusion ERwin DM のメタモデル情報を取得することができます。メタモデルにアクセスするには、AllFusion ERwin DM モデルへのアクセスと同じ方法を使用します。モデル データの場合と同じく、ISCSession::Open コールの ISCPersistenceUnit または ISCModelSet ポインタを使用して、作業中のモデル セットを指定します。

組み込みメタモデルは特別なケースになります。特定のメタデータ クラスの組み込みメタモデルを取得するには、ISCApplicationEnvironment インターフェイスの PropertyBag メソッドで作成した Property Bag コンポーネントを使用します。Property Bag インスタンスに、Application カテゴリの EMX Metadata Class または EM2 Metadata Class プロパティを追加して、アクセスする組み込みメタモデルのタイプを指定します。このインスタンスは、ISCPersistenceUnit または ISCModelSet ポインタの代わりに、ISCSession::Open コールの最初のパラメータとして送信する必要があります。ISCSession::Open コールの最初のパラメータが NULL の場合、永続ユニット内のトップ モデル セットの組み込みメタモデルである、EMX クラスのメタデータにアクセスします。

セッションがメタモデル情報にアクセスすることを指定するには、Open メソッドの Level パラメータを SCD_SL_M1 に設定します。

メタデータ構成の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCApplicationEnvironmentインターフェイス

次の表に、ISCApplicationEnvironment インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPropertyBag PropertyBag(VARIANT Category[省略可], VARIANT Name[省略可], VARIANT AsString[省略可])	Category と Name で指定された 1 つ以上のプロパティ値を持つプロパティ バッグを追加します。	Category: ■ VT_BSTR – 指定されたカテゴリの機能を返します。Application を指定する必要があります。 Name: ■ VT_BSTR – 指定された名前とカテゴリのプロパティを返します。EMX メタデータでは EMX Metadata Class、EM2 メタデータでは EM2 Metadata Class を指定する必要があります。 AsString: ■ Empty – プロパティ バッグのすべての値はネイティブ型になります。

ISCSessionインターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Open(IUnknown * Unit, VARIANT Level [省略可], VARIANT Flags [省略可])	Unit パラメータで特定された組み込みメタモデル、永続ユニット、またはモデル セットに自身をバインドします。	Unit: ■ NULL – 永続ユニットに含まれるトップ モデル セットの組み込みメタモデル。現在のバージョンでは、EMX クラス メタデータです。 ■ ISCPropertyBag – バッグの最初のプロパティ内のメタデータ クラスにより定義された組み込みメタモデル。 ■ ISCPersistenceUnit – 永続ユニットに含まれるトップ モデル セットのメタモデル。 ■ ISCModelSet – モデル セットのメタモデル。 Level: ■ SCD_SL_M1 – メタデータ アクセス。 Flags: ■ Empty – SCD_SF_NONE がデフォルトになります。

例 27

次の例は、C++で組み込みメタモデルにアクセスする方法を示します。例 1 の Application オブジェクトを使用します。

```
void AccessMetaModel( ISCApplicationPtr & scAppPtr )
{
ISCSessionPtr scSessionPtr = scAppPtr->GetSessions()->Add();   // 新規セッションを
// 追加
// EMX 組み込みメタモデルを開く
CComVariant varResult = scSessionPtr->Open(NULL, (long) SCD_SL_M1);  //   メタモデル レベル
if (varResult.vt == VT_BOOL && varResult.boolVal == FALSE)
return;
    // …
}
```

次の例は、Visual Basic .NET で組み込みメタモデルにアクセスする方法を示します。例 1 の Application オブジェクトを使用します。

```
Public Sub AccessMetaModel( ByRef scApp As SCAPI.Application )
Dim scBag As SCAPI.PropertyBag
Dim scSession As SCAPI.Session

' EM2 メタデータ クラスのプロパティ バッグを取得
scBag = scApp.ApplicationEnvironment.PropertyBag("Application ", "EM2 Metadata
Class")

' EM2 組み込みメタモデルを開く
scSession = scApp.Sessions.Add
scSession.Open( scBag, SCD_SL_M1 )
End Sub
```

API を閉じる

API クライアントがモデルへのアクセスを完了したら、開いているセッションを閉じて、永続ユニット コレクションを消去する必要があります。

ISCSessionインターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Close()	関連付けられた永続ユニットから自身の接続を解除します。	なし

ISCSessionCollectionインターフェイス

次の表に、ISCSessionCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Remove(VARIANT SessionId)	Session オブジェクトをコレクションから削除します。	SessionId: ■ VT_UNKNOWN – ISCSession インターフェイスへのポインタ指定されたセッションをコレクションから削除します。 ■ VT_I4 – セッション コレクション内のゼロベース インデックス。指定されたインデックスを持つセッションをコレクションから削除します。

例 28

次の例は、C++でセッションを閉じる方法を示します。Session オブジェクトが存在しており、セッションが開いていると仮定します。例 1 の Application オブジェクトを使用します。

```
void CloseSessions( ISCApplicationPtr & scAppPtr )
{
ISCSessionCollectionPtr scSessionColPtr = scAppPtr->GetSessions();
ISCSessionPtr scSessionPtr = scSessionColPtr->GetItem(COleVariant(0L))
// セッションを閉じる
scSessionPtr->Close();    // 1 つのセッションを閉じる
scSessionColPtr->Clear();   // セッションのコレクションを消去する
}
```

次の例は、Visual Basic .NET でセッションを閉じる方法を示します。Session オブジェクトが存在しており、セッションが開いていると仮定します。例 1 の Application オブジェクトを使用します。

```
Public Sub CloseSessions( scApp As SCAPI.Application )
Dim scSessionCol As SCAPI.Sessions
scSessionCol = scApp.Sessions
Dim scSession As SCAPI.Session

For Each scSession In scSessionCol
scSession.Close
Next
While (scSessionCol.Count > 0)
scSessionCol.Remove (0)
End
End Sub
```

永続ユニットを消去する

このセクションでは永続ユニットを消去する方法を説明します。

Persistence Units コレクションの永続ユニットが使用後に閉じられるかどうかは、アプリケーションのインスタンスが作成される状況によって決まります。クライアントが API をスタンドアロン モードで使用している場合、すべての永続ユニットが閉じます。クライアントが API をアドイン コンポーネントとして使用している場合、クライアント プログラムが終了しても永続ユニットは開いたままとなり、アプリケーションのユーザー インターフェイスから利用できます。ただし、プログラムの終了前に永続ユニット コレクションから永続ユニットを明示的に閉じるか、または削除した場合を除きます。

ISCPersistenceUnitCollection インターフェイス

次の表に、ISCPersistenceUnitCollection インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT_BOOL Clear()	すべてのユニットをコレクションから消去します。	なし

例 29

次の例は、C++で永続ユニットを消去する方法を示します。例 1 の Application オブジェクトが存在していると仮定します。

```
// 永続ユニットを削除
scAppPtr->GetPersistenceUnits()->Clear();
```

次の例は、Visual Basic .NET で永続ユニットを消去する方法を示します。例 1 の Application オブジェクトが存在していると仮定します。

```
scApp.PersistenceUnits.Clear
```

エラー処理

API では、汎用 COM エラー オブジェクトを使用してエラーを処理します。各プログラミング環境では、言語に固有のプロトコルによって汎用エラー オブジェクトからエラーを取得します。たとえば、C++および Visual Basic .NET では、例外処理を使用してエラーを処理します。アプリケーションを安定して動作させるために、API クライアントでエラー処理を使用して、潜在的なエラー（削除済みのオブジェクトや空のコレクションにアクセスしようとするなど）を補足することをお勧めします。

例 30

次の例は、C++でエラー処理を行う方法を示します。例 9 の Model Object オブジェクトが存在していると仮定します。

```
long GetObjectProperties(ISCModelObjectPtr & scObjPtr)
{
// プロパティのコレクションを取得
ISCModelPropertyCollectionPtr scPropColPtr;
try
     {
scPropColPtr = scObjPtr->GetProperties();
if (!scPropColPtr.GetInterfacePtr())
         {
AfxMessageBox("Unable to Get Properties Collection");
return FALSE;
         }
         // …
     }
catch(_com_error &error)
     {
AfxMessageBox(error.Description());
     }
}
```

次の例は、Visual Basic .NET でエラー処理を行う方法を示します。例 9 の Model Object オブジェクトが存在していると仮定します。

```
Public Sub GetObject(ByRef scSession As SCAPI.Session, ByRef objID As String)
Dim scObjCol as SCAPI.ModelObjects
Dim scObj as SCAPI.ModelObject

Try
scObjCol = scSession.ModelObjects
scObj = scObjCol.Item(objID)  ' 指定されたオブジェクト ID を持つオブジェクトを取得
Catch ex As Exception
' 失敗時
Console.WriteLine(" API Failed With Error message :"+ ex.Message())
End Try
End Sub
```

API では、汎用エラー オブジェクトだけでなく、拡張されたエラー処理メカニズムによって Application Environment Message ログを利用できます。このメッセージ ログで扱うことができる一連のメッセージは、トランザクションのような複雑な操作を確認する際にも役立ちます。

Application Environment Message ログ構成の詳細については、付録 A「API インターフェイスのリファレンス」の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCApplicationEnvironment

次の表に、ISCApplicationEnvironment インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
ISCPropertyBag PropertyBag(VARIANT Category[省略可], VARIANT Name[省略可], VARIANT AsString[省略可])	Category と Name で指定された 1 つ以上のプロパティ値を持つプロパティ バッグを追加します。	Category: VT_BSTR – Application.API を指定する必要があります。 Name: ■ VT_BSTR – 指定された名前とカテゴリのプロパティを返します。メッセージ ログにメッセージがあるかを確認するには、Is Empty を指定する必要があります。メッセージ ログの内容を取得するには、Log を指定する必要があります。 AsString: ■ Empty – プロパティ バッグのすべての値はネイティブ型になります。 ■ VT_BOOL – TRUE に設定すると、プロパティ バッグのすべての値は文字列型になります。

例 32

次の例は、C++で API を使用して、API 拡張メッセージ ログからメッセージを確認する方法を示します。例 1 の Application オブジェクトが存在していると仮定します。

```
CString GetExtendedErrorInfo(ISCApplicationPtr & scAppPtr)
{
CString csExtendedErrors = "";
long index = 0;

// ログにメッセージがあるかを確認
variant_t val = scAppPtr->GetApplicationEnvironment()->
GetPropertyBag("Application.Api.MessageLog","Is Empty")->
GetValue(COleVariant(index));

if (val.vt == VT_BOOL && val.boolVal == false)
     {
// ログの取得
val = m_scAppPtr->GetApplicationEnvironment()->
GetPropertyBag("Application.Api.MessageLog","Log")->
GetValue(COleVariant(index));
if (val.vt & VT_ARRAY)
         {
         // これは SAFEARRAY

         VARIANT HUGEP *pArray;
         HRESULT hr;

         // 配列要素へのポインタを取得
         hr = SafeArrayAccessData(val.parray, (void HUGEP**)&pArray);
         if (FAILED(hr))
             return csExtendedErrors;

         long numErrors = 0;
         VARIANT vValue = pArray[0];   // エラーの数
         if (vValue.vt == VT_I4)
              numErrors = vValue.lVal;

             // …
SafeArrayUnaccessData(val.parray);
         }
     }
}
```

次の例は、Visual Basic .NET で API を使用して、API 拡張メッセージ ログからメッセージを確認する方法を示します。例 1 の Application オブジェクトが存在していると仮定します。

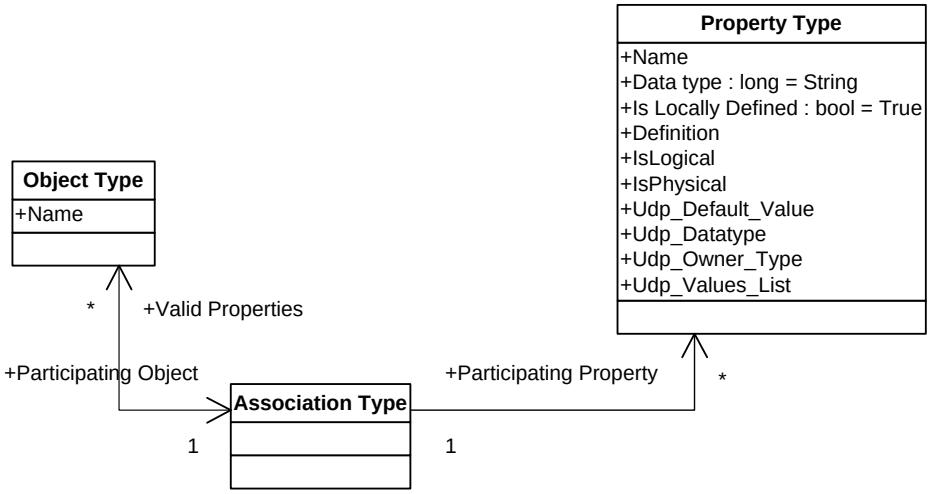
```
Public Sub GetExtendedErrorInfo( ByRef scApp As SCAPI.Application )
Dim nSize As Integer
Dim nWarnings As Integer
Dim nErrors As Integer
Dim nIdx As Integer
Dim nMsgNumber As Integer
Dim aErrors() As Object
' ログにメッセージがあるかを確認
If scApp.ApplicationEnvironment.PropertyBag("Application.Api.MessageLog", _ "Is
Empty").Value(0) = False Then
' ログの取得
aErrors = _
scApp.ApplicationEnvironment.PropertyBag("Application.Api.MessageLog", _
"Log").Value(0)
nSize = Int(aErrors(0))
nIdx = 1
nMsgNumber = 0
Do While nMsgNumber < nSize
Console.WriteLine("Error " & aErrors(nIdx) & "  " + aErrors(nIdx + 2))
Select Case aErrors(nIdx + 1)
Case SCAPI.SC_MessageLogSeverityLevels.SCD_ESL_WARNING
nWarnings = nWarnings + 1
Case SCAPI.SC_MessageLogSeverityLevels.SCD_ESL_ERROR
nErrors = nErrors + 1
End Select
nIdx = nIdx + 8
nMsgNumber = nMsgNumber + 1
Loop

Console.WriteLine("Total number of errors in the transaction " & Str(nSize) & "
with:" _& Str(nWarnings) & " warnings, " & Str(nErrors) & " errors.")

End If
End Sub
```

高度なタスク

このセクションでは、いくつかの高度なタスクとその実行方法について例を挙げて説明します。

ユーザー定義プロパティを作成する

ユーザー定義プロパティ(UDP)は、クライアントが AllFusion ERwin DM メタデータを拡張して、メタデータ レベルでオブジェクトを作成および変更する例の一つです。UDP 定義の構造は、すべてのネイティブ プロパティの定義と同様です。次のダイアグラムに、UDP 定義に関連するメタモデル オブジェクトを示します。



このダイアグラムでは、Property Type オブジェクトのインスタンスが UDP クラスを定義し、Object Type オブジェクトは、UDP が関連付けられるオブジェクト クラスを定義します。Association Type オブジェクトは、オブジェクトとプロパティ クラス間の関連を定義します。

UDP を定義するために必要なのは、Property Type オブジェクトのインスタンスを作成することだけです。残りの必要なデータは、AllFusion ERwin DM によって追加されます。次の表に、Property Type オブジェクトのプロパティおよびタグの説明を示します。

プロパティまたはタグの名前	説明	有効な引数
Name	プロパティ、UDP 名	AllFusion ERwin DM では、UDP に一意な名前を付けるために次の表記が使用されます。この表記では、ドット(.)記号で区切られた 3 つの部分から名前が構成されます。 <ObjectClassName>.<Logical/Physical>.<Name> この表記に基づく名前の例: Model.Logical.My UDP AllFusion ERwin DM のエディタ画面には、最後のコンポーネントのみが表示されます(上の例では My UDP)。
Datatype	プロパティ、SCVT_BSTR	このプロパティは読み取り専用です。AllFusion ERwin DM によって設定されます。すべての UDP 値は、文字列型のデータタイプです。
Is Locally Defined	プロパティ、TRUE	このプロパティは読み取り専用です。すべてのユーザー定義メタデータで TRUE に設定されます。
Definition	プロパティ、省略可	省略可能です。UDP の説明を表示するテキストです。
IsLogical	タグ、TRUE または FALSE	省略可能です。このタグは、論理モデリングの UDP で TRUE になります。

プロパティまたはタグの名前	説明	有効な引数
IsPhysical	タグ、TRUE または FALSE	省略可能です。このタグは、物理モデリングの UDP で TRUE になります。
Udp_Default_Value	タグ	省略可能です。UDP のデフォルト値となる文字列です。
Udp_Datatype	タグ	AllFusion ERwin DM エディタでの UDP 値の解釈を定義します。有効な値は次のとおりです: ■ 1 (Integer) ■ 2 (Text) ■ 3 (Date) ■ 4 (Command) ■ 5 (Real) ■ 6 (List) 可能なプロパティ値は次のとおりです: ■ VT_I4 – 上の一覧の数値を使用します。 ■ VT_BSTR – 上の一覧の文字列値を使用します。 指定しない場合は、Text タイプになります。
Udp_Owner_Type	タグ	必須です。UDP のホスト インスタンスに対するオブジェクト クラスを定義します。 ■ VT_BSTR – オブジェクト クラス名。指定されたクラス名でホスト クラスを示します。 ■ VT_BSTR – オブジェクト クラスの Class ID。指定された識別子でホスト クラスを示します。
Udp_Values_List	タグ	カンマ区切りの文字列値。この一覧の値のみが有効な UDP になります。 Udp_Datatype タグが List に設定されている場合のみ、有効になります。

例 33

次の例は、Visual Basic .NET で API を使用して UDP を定義する方法を示します。

```
Public Sub Main()
Dim oAPI As New SCAPI.Application
Dim oPU As SCAPI.PersistenceUnit
Dim oSession As SCAPI.Session

' すべてデフォルトで新規モデルを作成
oPU = oAPI.PersistenceUnits.Create(Nothing)

' 新規セッションの追加
oSession = oAPI.Sessions.Add

' モデルのトップ（EMX）モデル セットでメタデータ レベルのセッションを開始
oSession.Open(oPU, SCAPI.SC_SessionLevel.SCD_SL_M1)

' トランザクションの開始
Dim TransId As Object
TransId = oSession.BeginNamedTransaction("Create UDP")

' 新規 UDP 定義の作成
Dim oUDP As SCAPI.ModelObject
oUDP = oSession.ModelObjects.Add("Property Type")

' プロパティの追加
oUDP.Properties("Name").Value = "Entity.Logical.My UDP"
oUDP.Properties("Udp_Owner_Type").Value = "Entity"
oUDP.Properties("IsLogical").Value = True
oUDP.Properties("Udp_Datatype").Value = "Text"

' 変更のコミット
oSession.CommitTransaction(TransId)

' 結果のレポート
Dim oProperty As SCAPI.ModelProperty

Console.WriteLine(" --- Report the new User-Defined Property properties")
For Each oProperty In oUDP.Properties
Console.WriteLine(oProperty.ClassName + " '" + oProperty.FormatAsString + "'")
Next

Console.WriteLine(" Press Entry to exit ")
Console.Read()
' セッションの解放
oSession.Close()
oSession = Nothing
oAPI.Sessions.Clear()

' ファイルに保存
oPU.Save("w:\UDP.erwin", "OVF=Yes")

End Sub
```

履歴の追跡

モデル、エンティティ、属性、テーブル、およびカラムの履歴情報を保存できます。AllFusion ERwin DM では、History オブジェクトを使用して履歴情報をモデルに保存します。

API の機能を使用して、History オブジェクトを直接操作することなく、履歴の追跡プロセスをカスタマイズすることができます。ISCSession インターフェイスの BeginNamedTransaction 関数は、履歴の追跡プロパティが追加された Property Bag インスタンスを受け入れます。このプロパティは、外部トランザクションの開始時に有効となり、トランザクションのスコープに制限されます。

ISCSession インターフェイス

次の表に、ISCSession インターフェイスの情報を示します。

シグネチャ	説明	有効な引数
VARIANT BeginNamedTransaction(BSTR Name, VARIANT PropertyBag [省略可])	指定された名前のセッションにトランザクションを開きます。トランザクションの識別子を返します。	Name – 新規トランザクションの名前。 PropertyBag – トランザクションの履歴を追跡するパラメータのコレクション。

次の表に、新規モデルの作成に使用するプロパティを説明します。

プロパティ名	タイプ	説明
History Tracking	ブール値	TRUE – トランザクション中に生成されたすべての履歴情報が、API イベントとしてマークされます。TRUE は、このプロパティを指定しない場合のデフォルト値です。 FALSE – 標準の AllFusion ERwin DM メカニズムを使用して履歴を追跡します。
History Description	BSTR	History Tracking プロパティが TRUE の場合、履歴イベントの Description フィールドの内容を表します。

注: すべてのプロパティー覧については、付録 A「API インターフェイスのリファレンス」を参照してください。

例 34

次の例は、Visual Basic .NET で、エンティティと属性の履歴レコードを API イベントとしてマークする方法、および API の History Tracking の説明を含む履歴レコードをマークする方法を示します。

```
Public Sub Main()
Sub Main()
Dim oApi As New SCAPI.Application
Dim oBag As New SCAPI.PropertyBag
Dim oPU As SCAPI.PersistenceUnit

' 新しい論理/物理モデルを構築し、残りはデフォルト値を受け入れる
oBag.Add("Model Type", "Combined")
oPU = oApi.PersistenceUnits.Create(oBag)

' 今後の再利用のためにバッグを消去する
oBag.ClearAll()

' セッションの開始
Dim oSession As SCAPI.Session

oSession = oApi.Sessions.Add
oSession.Open(oPU)

' トランザクション プロパティを含むプロパティ バッグを用意
oBag.Add("History Description", "API History Tracking")

' トランザクションの開始
Dim nTransId As Object

nTransId = oSession.BeginNamedTransaction("Create Entity and Attribute", oBag)

' エンティティと属性の作成
Dim oEntity As SCAPI.ModelObject
Dim oAttribute As SCAPI.ModelObject

oEntity = oSession.ModelObjects.Add("Entity")
oAttribute = oSession.ModelObjects.Collect(oEntity).Add("Attribute")
oAttribute.Properties("Name").Value = "Attr A"

' コミット
oSession.CommitTransaction(nTransId)

End Sub
```

履歴を保存したいモデル オブジェクトの履歴オプションを選択したり、追跡するイベント タイプを制御することができます。これは、[モデル プロパティ]ダイアログ ボックスの[履歴オプション]タブで設定できます。

[オブジェクトの履歴を保存するタイミング]の各チェック ボックスをオフにすると、その API カテゴリの履歴イベントは記録されません。これらのチェック ボックスや、[履歴の保存]グループの各モデル オブジェクト タイプのチェックボックスの状態を API から制御するには、これらのチェック ボックスの状態が保存されているモデル内のプロパティ値を設定します。

付録A: API インターフェイスのリファレンス

この付録では、API に含まれるインターフェイスと、これらのインターフェイスに関連するメソッドと引数の一覧を示します。また、列挙体の情報を含むセクションや、さまざまな Property Bag コンポーネントを説明したセクションもあります。

本章には次のトピックがあります。

API インターフェイス

このセクションでは、各 API インターフェイスと、これらのインターフェイスに関連するメソッドについて説明します。必要に応じて、シグネチャと有効な引数についても説明します。

注: 一部のパラメータには、[省略可]という指定があります。これは、任意指定のパラメータであることを示します。

ISCApplication

ISCApplication インターフェイスは、API クライアントのエントリ ポイントです。このコンポーネントのインスタンスを外部から 1 つだけ作成して、API をアクティブにすることができます。クライアントは、インターフェイスのプロパティとメソッドを使用して、このインターフェイス階層を移動し、他の API 機能にアクセスできます。

次の表に、ISCApplication インターフェイスのメソッドを示します。

メソッド	説明
BSTR ApiVersion()	API のバージョン。
ISCApplicationEnvironment * ApplicationEnvironment()	ランタイム環境の属性と利用可能な機能をレポートします(たとえば、アドイン モード、ユーザー インターフェイスの表示設定など)。
ISCApplicationServiceCollection * ApplicationServices()	さまざまなアプリケーション サービスにアクセスできます(たとえば、フォワード エンジニアリング、リバース エンジニアリング、完全比較など)。現行の API バージョンでは利用できません。
ISCModelDirectoryCollection * ModelDirectories()	現在のマシンからアクセス可能なモデル ディレクトリを収集します。
BSTR Name()	モデリング ツール アプリケーションの名前
ISCPersistenceUnitCollection * PersistenceUnits()	アプリケーションに読み込まれたすべての永続ユニットのコレクションを返します。

メソッド	説明
ISCSessionCollection * Sessions()	アプリケーション内に作成されたセッションのコレクションを返します。
BSTR Version()	モデリング ツール アプリケーションのバージョン

ISCApplicationEnvironment

ISCApplicationEnvironment インターフェイスには、ランタイム環境についての情報が含まれています。

次の表に、ISCApplicationEnvironment インターフェイスのメソッドを示します。

メソッド	説明
ISCPropertyBag * PropertyBag(VARIANT Category [省略可], VARIANT Name [省略可], VARIANT AsString [省略可])	Category と Name で指定された 1 つ以上のプロパティ値を含むプロパティ バッグを追加します。 詳細については「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCApplicationEnvironment::PropertyBag の引数

PropertyBag 関数のシグネチャを示します:

```
ISCPropertyBag *PropertyBag(VARIANT Category, VARIANT Name, VARIANT AsString)
```

次の表に、PropertyBag 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Category [省略可]	Empty	すべてのカテゴリの全機能セットを返します。
Category [省略可]	VT_BSTR – カテゴリ名	指定されたカテゴリの機能を返します。 カテゴリ名の詳細については、この付録の「アプリケーション環境のプロパティ バグ」セクションを参照してください。
Name [省略可]	Empty	選択したカテゴリのすべてのプロパティを返します。
Name [省略可]	VT_BSTR – プロパティ名	指定された名前とカテゴリを持つプロパティを返します。 プロパティ名の詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
AsString [省略可]	Empty	プロパティ バッグのすべての値はネイティブ型になります。

パラメータ	有効なタイプ/値	説明
AsString [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、プロパティ バッグのすべての値は文字列型になります。

ISCModelDirectory

Model Directory は、1 つのモデル ディレクトリ エントリ上の情報をカプセル化します。Model Directory の例は、ファイル システム上のディレクトリや、AllFusion Model Manager ライブラリです。

次の表に、ISCModelDirectory インターフェイスのメソッドを示します。

メソッド	説明
VARIANT_BOOL CopyDirectory(BSTR Source, BSTR Destination, VARIANT Disposition [省略可], VARIANT Overwrite [省略可])	1 つ以上のモデル ディレクトリを別の場所へ再帰的にコピーします。
VARIANT_BOOL CopyDirectoryUnit(BSTR Source, BSTR Destination, VARIANT Disposition [省略可], VARIANT Overwrite [省略可])	1 つ以上のモデル ディレクトリ ユニットを別の場所にコピーします。
VARIANT_BOOL CreateDirectory(BSTR Name)	モデル ディレクトリを作成します。
VARIANT_BOOL DeleteDirectory(BSTR Locator, VARIANT Force [省略可])	指定されたモデル ディレクトリとその内容を削除します。
VARIANT_BOOL DeleteDirectoryUnit(BSTR Locator, VARIANT Force [省略可])	指定されたモデル ディレクトリ ユニットを削除します。
VARIANT_BOOL DirectoryExists(BSTR Locator)	指定されたディレクトリが存在する場合、TRUE を返します。
VARIANT_BOOL DirectoryUnitExists(BSTR Locator)	指定されたディレクトリ ユニットが存在する場合、TRUE を返します。
SC_ModelDirectoryFlags Flags()	Model Directory のフラグ。32 ビットのプロパティ フラグ ワードです。
VARIANT_BOOL IsOfType(ISCModelDirectory * Directory)	ディレクトリの接続タイプが自身と同じ場合、TRUE を返します。 たとえば、ディレクトリ エントリが同じ Model Manager データベースに存在し、同じログイン属性（ユーザー名、パスワードなど）を持つ場合、同一タイプであると判定されます。
ISCModelDirectory * LocateDirectory (BSTR Locator, VARIANT Filter [省略可])	ディレクトリのサブエントリの列挙を開始します。
ISCModelDirectory * LocateDirectoryNext()	ディレクトリの列挙で、次のサブエントリを検索します。モデル ディレクトリ エントリがそれ以上見つからない場合は、NULL ポインタを返します。
ISCModelDirectoryUnit * LocateDirectoryUnit (BSTR Locator, VARIANT Filter [省略可])	ディレクトリ ユニットの列挙を開始します。

メソッド	説明
ISCModelDirectoryUnit * LocateDirectoryUnitNext()	ディレクトリの列挙で、次のユニットを検索します。
BSTR Locator()	絶対パスとパラメータを含むディレクトリの場所。パスワード情報は含まれません。
VARIANT_BOOL MoveDirectory(BSTR Source, BSTR Destination, VARIANT Disposition [省略可], VARIANT Overwrite [省略可])	1 つ以上のディレクトリを別の場所に再帰的に移動します。
VARIANT_BOOL MoveDirectoryUnit(BSTR Source, BSTR Destination, VARIANT Disposition [省略可], VARIANT Overwrite [省略可])	1 つ以上のモデル ディレクトリ ユニットを別の場所に移動します。
BSTR Name()	Model Directory の名前。たとえば、パス情報を含まないファイル システム ディレクトリ名です。
ISCPropertyBag* PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可])	ディレクトリ プロパティを含むプロパティ バッグのポインタを返します。 **注:** ディレクトリ プロパティがプロパティ バッグに含まれるのは、値を持つ場合のみです。プロパティに値セットがない場合、プロパティ バッグにはリストされません。
void PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可], ISCPropertyBag* Property Bag)	ディレクトリ プロパティを含むプロパティ バッグのポインタを受け入れます。
SC_ModelDirectoryType Type()	ディレクトリのタイプ。

ISCModelDirectory::CopyDirectory の引数

CopyDirectory 関数のシグネチャを示します:

```
VARIANT_BOOL CopyDirectory(BSTR Source, BSTR Destination, VARIANT Disposition,
VARIANT Overwrite)
```

次の表に、CopyDirectory 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Source	BSTR – ディレクトリ パスの文字列	ディレクトリ パスを表します。最後のパス部分にワイルドカード文字を含めて、コピー元となる 1 つ以上のモデル ディレクトリの場所を指定できます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。
Source	Empty	自身を操作の対象に指定します。
Destination	BSTR – ディレクトリ パスの文字列	相対または絶対パスを表す文字列です。モデル ディレクトリのコピー先の場所を指定します。ワイルドカード文字は使用できません。

パラメータ	有効なタイプ/値	説明
Disposition [省略可]	Empty	
Disposition [省略可]	VT_BSTR – セミコロン区切りのキーワード パラメータ	コピー先のアクセス属性（セッション再開など）を指定します。絶対パスの指定と組み合わせることで、Source と Destination に異なる Model Manager データベースを指定できます。
Overwrite [省略可]	Empty	既存のディレクトリを上書きするかどうかを指定します。デフォルトでは、ディレクトリは上書きされます。
Overwrite [省略可]	VT_BOOL - TRUE または FALSE	既存のディレクトリを上書きするかどうかを指定します。TRUE の場合、ディレクトリは上書きされます。

ISCModelDirectory::CopyDirectoryUnit の引数

CopyDirectoryUnit 関数のシグネチャを示します:

```
VARIANT_BOOL CopyDirectoryUnit(BSTR Source, BSTR Destination, VARIANT
Disposition, VARIANT Overwrite)
```

次の表に、CopyDirectoryUnit 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Source	BSTR – ディレクトリ パスの文字列	ディレクトリ パスを表します。最後のパス部分にワイルドカード文字を含めて、コピー元となる 1 つ以上のモデル ディレクトリ ユニットの場所を指定できます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。
Source	Empty	自身を操作の対象に指定します。
Destination	BSTR – ディレクトリ パスの文字列	相対または絶対パスを表す文字列です。モデル ディレクトリのコピー先の場所を指定します。ワイルドカード文字は使用できません。
Disposition [省略可]	Empty	
Disposition [省略可]	VT_BSTR – セミコロン区切りのキーワード パラメータ	コピー先のアクセス属性（セッション再開など）を指定します。絶対パスの指定と組み合わせることで、Source と Destination に異なる Model Manager データベースを指定できます。

パラメータ	有効なタイプ/値	説明
Overwrite [省略可]	Empty	既存のディレクトリ ユニットを上書きするかどうかを指定します。デフォルトでは、ディレクトリ ユニットは上書きされます。
Overwrite [省略可]	VT_BOOL - TRUE または FALSE	既存のディレクトリ ユニットを上書きするかどうかを指定します。TRUE の場合、ディレクトリ ユニットは上書きされます。

ISCModelDirectory::CreateDirectory の引数

CreateDirectory 関数のシグネチャを示します:

```
VARIANT_BOOL CreateDirectory( BSTR Name)
```

次の表に、CreateDirectory 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Name	BSTR – ディレクトリ名の文字列	作成するディレクトリの名前を指定します。

ISCModelDirectory::DeleteDirectory の引数

DeleteDirectory 関数のシグネチャを示します:

```
VARIANT_BOOL DeleteDirectory( BSTR Locator, VARIANT Force)
```

次の表に、DeleteDirectory 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	BSTR – ディレクトリ名の文字列	ディレクトリ パスを指定します。パスの最後の部分にワイルドカード文字を含めることができます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。
Locator	Empty	自身を操作の対象に指定します。
Force [省略可]	Empty	ファイル システムのロック(読み取り専用など)を上書きしません。
Force [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、ファイル システムのロック(読み取り専用など)を上書きします。

ISCModelDirectory::DeleteDirectoryUnit の引数

DeleteDirectoryUnit 関数のシグネチャを示します:

```
VARIANT_BOOL DeleteDirectoryUnit( BSTR Locator, VARIANT Force)
```

次の表に、DeleteDirectoryUnit 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	BSTR – ディレクトリ名の文字列	ディレクトリ ユニットのパスを指定します。パスの最後の部分にワイルドカード文字を含めることができます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。
Force [省略可]	Empty	ファイル システムのロック設定を上書きしません。
Force [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、ファイル システムの読み取り専用のようなロックを上書きします。

ISCModelDirectory::DirectoryExists の引数

DirectoryExists 関数のシグネチャを示します:

```
VARIANT_BOOL DirectoryExists( BSTR Locator)
```

次の表に、DirectoryExists 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	BSTR – ディレクトリ名の文字列	ディレクトリ パスを指定します。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。

ISCModelDirectory::DirectoryUnitExists の引数

DirectoryUnitExists 関数のシグネチャを示します:

```
VARIANT_BOOL DirectoryUnitExists( BSTR Locator)
```

次の表に、DirectoryUnitExists 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	BSTR – ディレクトリ名の文字列	ディレクトリ ユニットのパスを指定します。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。

ISCModelDirectory::IsOfType の引数

IsOfType 関数のシグネチャを示します:

```
VARIANT_BOOL IsOfType(ISCModelDirectory * Directory)
```

次の表に、IsOfType 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Directory	ISCModelDirectory *.Model Directory コンポーネントのポインタ	ディレクトリを指定します。

ISCModelDirectory::LocateDirectory の引数

LocateDirectory 関数のシグネチャを示します:

```
ISCModelDirectory * LocateDirectory (BSTR Locator, VARIANT Filter)
```

次の表に、LocateDirectory 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	BSTR – ディレクトリの場所を表す文字列	ディレクトリ パスを指定します。パスの最後の部分にワイルドカード文字を含めて、サブエントリを検索できます。 このパスで正確な場所を指定すると、単一のモデル ディレクトリ エントリを返すために使用することもできます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。

パラメータ	有効なタイプ/値	説明
Filter [省略可]	VT_BSTR – オプション	検索範囲を絞るためのオプション セットを指定します。

ISCModelDirectory::LocateDirectoryUnit の引数

LocateDirectoryUnit 関数のシグネチャを示します:

```
ISCModelDirectoryUnit * LocateDirectoryUnit (BSTR Locator, VARIANT Filter)
```

次の表に、LocateDirectoryUnit 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	BSTR – ディレクトリまたはユニットの場所を表す文字列	ディレクトリ パスを指定します。パスの最後の部分にワイルドカード文字を含めてユニットを検索できます。 このパスで正確な場所を指定すると、単一のモデル ディレクトリ ユニットを返すために使用することもできます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。
Filter [省略可]	VT_BSTR – オプション	検索範囲を絞るためのオプション セットを指定します。

ISCModelDirectory::MoveDirectory の引数

MoveDirectory 関数のシグネチャを示します:

```
VARIANT_BOOL MoveDirectory(BSTR Source, BSTR Destination, VARIANT Disposition,
VARIANT Overwrite)
```

次の表に、MoveDirectory 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Source	BSTR – ディレクトリ パスの文字列	ディレクトリ パスを指定します。最後のパス 部分にワイルドカード文字を含めて、移動する 1 つ以上のモデル ディレクトリの場所を指定できます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。
Source	Empty	自身を操作の対象に指定します。

パラメータ	有効なタイプ/値	説明
Destination	BSTR – ディレクトリ パスの文字列	相対または絶対パスを表す文字列です。モデル ディレクトリの移動先の場所を指定します。ワイルドカード文字は使用できません。
Disposition [省略可]	Empty	
Disposition [省略可]	VT_BSTR – セミコロン区切りのキーワード パラメータ	移動先のアクセス属性（セッション再開など）を指定します。絶対パスの指定と組み合わせることで、Source と Destination に異なる Model Manager データベースを指定できます。
Overwrite [省略可]	Empty	既存のディレクトリを上書きするかどうかを指定します。デフォルトでは、ディレクトリは上書きされます。
Overwrite [省略可]	VT_BOOL – TRUE または FALSE	既存のディレクトリを上書きするかどうかを指定します。TRUE の場合、ディレクトリは上書きされます。

ISCModelDirectory::MoveDirectoryUnit の引数

MoveDirectoryUnit 関数のシグネチャを示します:

```
VARIANT_BOOL MoveDirectoryUnit(BSTR Source, BSTR Destination, VARIANT
Disposition, VARIANT Overwrite)
```

次の表に、MoveDirectoryUnit 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Source	BSTR – ディレクトリ パスの文字列	ディレクトリ パスを表します。最後のパス コンポーネントにワイルドカード文字を含めて、移動する 1 つ以上のモデル ディレクトリ ユニットの場所を指定できます。 絶対パスを指定すると、AllFusion Model Manager のデータベース情報とアクセス パラメータは無視されます。
Source	Empty	自身を操作の対象に指定します。
Destination	BSTR – ディレクトリ パスの文字列	相対または絶対パスを表す文字列です。モデル ディレクトリ ユニットの移動先の場所を指定します。ワイルドカード文字は使用できません。
Disposition [省略可]	Empty	

パラメータ	有効なタイプ/値	説明
Disposition [省略可]	VT_BSTR – セミコロン区切りのキーワード パラメータ	移動先のアクセス属性（セッション再開など）を指定します。絶対パスの指定と組み合わせることで、Source と Destination に異なる Model Manager データベースを指定できます。
Overwrite [省略可]	Empty	既存のモデル ディレクトリ ユニットを上書きするかどうかを指定します。デフォルトでは、モデル ディレクトリ ユニットは上書きされます。
Overwrite [省略可]	VT_BOOL – TRUE または FALSE	既存のモデル ディレクトリ ユニットを上書きするかどうかを指定します。TRUE の場合、モデル ディレクトリ ユニットは上書きされます。

ISCModelDirectory::PropertyBag の引数（Get 関数）

PropertyBag (Get)関数のシグネチャを示します:

```
ISCPropertyBag * PropertyBag(VARIANT List, VARIANT AsString)
```

次の表に、PropertyBag (Get)関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]	VT_BSTR – セミコロンで区切られたプロパティ名の一覧	モデル ディレクトリ プロパティの一覧を指定します。一覧に指定されたプロパティのみが、返されるプロパティ バッグに配置されます。 有効なプロパティ名の詳細については、この付録の「モデル ディレクトリとモデル ディレクトリ ユニットのプロパティ バッグ」セクションを参照してください。
List [省略可]	Empty	すべてのプロパティ セットを要求します。
AsString [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、プロパティ バッグのすべての値が文字列型になります。デフォルトは FALSE で、すべての値はそれぞれのネイティブ型になります。
AsString [省略可]	Empty	プロパティ バッグのすべての値はネイティブ型になります。

ISCModelDirectory::PropertyBag の引数（Set 関数）

PropertyBag (Set)関数のシグネチャを示します:

```
void PropertyBag(VARIANT List, VARIANT AsString, ISCPropertyBag * propBag)
```

次の表に、PropertyBag (Set)関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]		未使用
AsString [省略可]		未使用
propBag	ISCPropertyBag *	処理対象のディレクトリ プロパティを含むプロパティ バッグのポインタ。 有効なプロパティ名とフォーマットの詳細については、この付録の「モデル ディレクトリとモデル ディレクトリ ユニットのプロパティ バッグ」セクションを参照してください。

ISCModelDirectoryCollection

Model Directory Collection は、利用可能なすべてのトップレベル モデル ディレクトリの一覧で、その中の 1 つをアプリケーション ユーザー インターフェイスで利用することができます。クライアントは、新規モデル ディレクトリをこのコレクションに登録することができます。

メソッド	説明
IUnknown _NewEnum()	コレクション列挙子オブジェクトのインスタンスを作成します。
ISCModelDirectory * Add(BSTR Locator, VARIANT Disposition [省略可])	新規のトップレベル ディレクトリを、利用可能なディレクトリの一覧に追加します。
VARIANT_BOOL Clear()	すべてのトップレベル ディレクトリをコレクションから削除し、関連する Model Manager データベースからディレクトリを切断します。
long Count()	コレクション内の ModelDirectory コンポーネントの数。
ISCModelObject * Item(long nIndex)	順序位置によって指定された IUnknown インターフェイスのポインタを返します。
VARIANT_BOOL Remove(VARIANT Selector, VARIANT_BOOL Disconnect [省略可])	トップレベル ディレクトリを、利用可能なディレクトリの一覧から削除します。

ISCModelDirectoryCollection::Add の引数

Add 関数のシグネチャを示します:

```
ISCModelDirectory * Add(BSTR Locator, VARIANT Disposition)
```

次の表に、Add 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	BSTR – モデル ディレクトリの場所	モデル ディレクトリの場所と、ストレージへのアクセスに必要な属性を指定します。
Disposition [省略可]	VT_BSTR – キーワード パラメータの一覧	アクセス属性(セッション再開など)を指定します。

ISCModelDirectoryCollection::Item の引数

Item 関数のシグネチャを示します:

```
ISCModelDirectory * Item(long nIndex)
```

次の表に、Item 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
nIndex	Long 型の数値	Model Directory 項目の順序位置を指定します。ゼロベース インデックスです。
Class [省略可]	Empty	nIndex で指定されたオブジェクトを返します。

ISCModelDirectoryCollection::Remove の引数

Remove 関数のシグネチャを示します:

```
VARIANT_BOOL Remove(VARIANT Selector, VARIANT_BOOL Disconnect [省略可])
```

次の表に、Remove 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Selector	VT_UNKNOWN – ISCModelDirectory ポインタ	削除する Model Directory のオブジェクト ポインタ。
Selector	VT_I4 – 数値インデックス	ゼロベース インデックスを使用して、削除するモデル ディレクトリを指定します。

ISCModelDirectoryUnit

Model Directory Unit は、1 つのディレクトリ ユニット上の情報をカプセル化します。ファイル システムのファイルや AllFusion Model Manager モデルは、Model Directory Unit の一例です。

次の表に、ISCModelDirectoryUnit インターフェイスのメソッドを示します。

メソッド	説明
SC_ModelDirectoryFlags Flags()	モデル ディレクトリ ユニットのフラグ。32 ビットのプロパティ フラグ ワードです。 Model Directory フラグの詳細については、この付録の「列挙体」セクションを参照してください。
VARIANT_BOOL IsOfType(ISCModelDirectory * Directory)	ディレクトリの接続タイプが自身と同じ場合、TRUE を返します。 たとえば、ディレクトリ エントリが同じ Model Manager データベースに存在し、同じログイン属性（ユーザー名、パスワードなど）を持つ場合、同一タイプであると判定されます。
BSTR Locator()	絶対パスとパラメータを含むディレクトリ ユニットの場所。パスワード情報は含まれません。
BSTR Name()	モデル ディレクトリ ユニットの名前。たとえば、パス情報なしのファイル システム ファイル名です。
ISCPropertyBag* PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可])	ディレクトリ ユニットのプロパティを含むプロパティ バッグのポインタを返します。 **注:** ディレクトリ ユニットのプロパティがプロパティ バッグに含まれるのは、値を持つ場合のみです。プロパティに値セットがない場合、プロパティ バッグにはリストされません。
void PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可], ISCPropertyBag* Property Bag)	ディレクトリ ユニットのプロパティを含むプロパティ バッグのポインタを受け入れます。
SC_ModelDirectoryType Type()	ディレクトリのタイプ。

ISCModelDirectoryUnit::IsOfType の引数

IsOfType 関数のシグネチャを示します:

```
VARIANT_BOOL IsOfType(ISCModelDirectory * Directory)
```

次の表に、IsOfType 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Directory	ISCModelDirectory * – Model Directory コンポーネントのポインタ	ディレクトリを指定します。

ISCModelDirectoryUnit::PropertyBag の引数（Get 関数）

PropertyBag（Get）関数のシグネチャを示します:

```
ISCPropertyBag * PropertyBag(VARIANT List, VARIANT AsString)
```

次の表に、PropertyBag (Get)関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]	VT_BSTR – セミコロンで区切られたプロパティ名の一覧	モデル ディレクトリ ユニットのプロパティ一覧を指定します。一覧に指定されたプロパティのみが、返されるプロパティ バッグ内に配置されます。 有効なプロパティ名の詳細については、この付録の「モデル ディレクトリとモデル ディレクトリ ユニットのプロパティ バッグ」セクションを参照してください。
List [省略可]	Empty	すべてのプロパティ セットを要求します。
AsString [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、プロパティ バッグのすべての値が文字列型になります。デフォルトは FALSE で、すべての値はそれぞれのネイティブ型になります。
AsString [省略可]	Empty	プロパティ バッグのすべての値はネイティブ型になります。

ISCModelDirectoryUnit::PropertyBag の引数（Set 関数）

PropertyBag（Set）関数のシグネチャを示します:

```
void PropertyBag(VARIANT List, VARIANT AsString, ISCPropertyBag * propBag)
```

次の表に、PropertyBag (Set)関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]		未使用
AsString [省略可]		未使用
propBag	ISCPropertyBag *	処理対象のユニット プロパティを含むプロパティ バッグのポインタ。 有効なプロパティ名とフォーマットの詳細については、この付録の「モデル ディレクトリとモデル ディレクトリ ユニットのプロパティ バッグ」セクションを参照してください。

ISCModelObject

ISCModelObject インターフェイスは、モデル内のオブジェクトを表します。

次の表に、ISCModelObject インターフェイスのメソッドを示します。

メソッド	説明
SC_ModelObjectFlags Flags()	オブジェクトのフラグを返します。 SC_ModelObjectFlags の詳細については、この付録の「列挙体」セクションを参照してください。
SC_CLSID ClassId()	現在のオブジェクトのクラス識別子を返します。
BSTR ClassName()	現在のオブジェクトのクラス名を返します。
ISCModelPropertyCollection * CollectProperties(VARIANT ClassIds [省略可], VARIANT MustBeOn [省略可], VARIANT MustBeOff [省略可])	必要なタイプのプロパティ コレクションを返します。このメソッドは、コレクションが空の場合でも有効なコレクションを常に返します。
ISCModelObject * Context()	モデルのオブジェクト ツリーにあるオブジェクトのコンテキスト（親）を返します。現在のオブジェクトがツリーのルートである場合は NULL を返します。
VARIANT_BOOL IsInstanceOf(VARIANT ClassId)	自身が受け渡されたクラスのインスタンスである場合、TRUE を返します。このメソッドは継承を反映します。ClassId に上位クラスが含まれる場合は、このメソッドは TRUE を返します。
VARIANT_BOOL IsValid()	自分自身が有効な場合、TRUE を返します。このメソッドを使用して、参照先オブジェクトが削除されているかを検出します。
BSTR Name()	現在のオブジェクトの名前または文字列識別子を返します。
SC_OBJID ObjectId()	現在のオブジェクトを一意に識別します。
ISCModelPropertyCollection * Properties()	利用可能なすべてのプロパティのプロパティ コレクションを返します。

ISCModelObject::CollectProperties の引数

CollectProperties 関数のシグネチャを示します:

```
ISCModelPropertyCollection * CollectProperties(VARIANT ClassIds, VARIANT
MustBeOn, VARIANT MustBeOff)
```

次の表に、CollectProperties 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ClassIds [省略可]	Empty	オブジェクトのすべてのプロパティを返します。

パラメータ	有効なタイプ/値	説明
ClassIds [省略可]	VT_ARRAY|VT_BSTR – プロパティ ID の SAFEARRAY	プロパティ クラス識別子の一覧を指定します。 有効なプロパティ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassIds [省略可]	VT_ARRAY|VT_BSTR – プロパティ名の SAFEARRAY	プロパティ クラス名の一覧を指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassIds [省略可]	VT_BSTR – プロパティ ID	プロパティ クラスを指定します。 有効なプロパティ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassIds [省略可]	VT_BSTR – プロパティ名	プロパティ クラスを指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassIds [省略可]	VT_BSTR – セミコロンで区切られた ID の一覧	プロパティ クラス識別子の一覧を指定します。 有効な ID の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassIds [省略可]	VT_BSTR – セミコロンで区切られたプロパティ名の一覧	プロパティ クラス名の一覧を指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
MustBeOn [省略可]	Empty	SCD_MPF_DONT_CARE がデフォルトになり、フィルタは適用されません。
MustBeOn [省略可]	VT_I4 – オンにする必要がある SC_ModelObjectFlags フラグ	指定されたフラグ セットを持つプロパティを特定します。 SC_ModelObjectFlags の詳細については、この付録の「列挙体」セクションを参照してください。
MustBeOff [省略可]	Empty	SCD_MPF_DONT_CARE がデフォルトになり、フィルタは適用されません。
MustBeOff [省略可]	VT_I4 – オフにする必要がある SC_ModelObjectFlags フラグ	指定されたフラグを持たないプロパティを特定します。 SC_ModelObjectFlags の詳細については、この付録の「列挙体」セクションを参照してください。

ISCModelObject::IsInstanceOf の引数

IsInstanceOf 関数のシグネチャを示します:

```
VARIANT_BOOL IsInstanceOf(VARIANT ClassId)
```

次の表に、IsInstanceOf 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ClassId	VT_BSTR – オブジェクト クラスの ID	指定された識別子で対象オブジェクト クラスを特定します。 有効なクラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId	VT_BSTR – オブジェクト クラス名	指定された名前でオブジェクト クラスを特定します。 有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCModelObjectCollection

ISCModelObjectCollection インターフェイスは、アクティブなセッションに接続されているモデル内のオブジェクトのコレクションです。フィルタ条件を設定して、このコレクションのメンバを制限することができます。

次の表に、ISCModelObjectCollection インターフェイスのメソッドを示します。

メソッド	説明
IUnknown _NewEnum()	コレクション列挙子オブジェクトのインスタンスを作成します。
ISCModelObject * Add(VARIANT Class, VARIANT ObjectId)	Class タイプのオブジェクトをモデルに追加します。
SC_CLSID * ClassIds()	クラス識別子(オブジェクト タイプ ID など)の SAFEARRAY を返します。 このコレクションの作成時にコレクションのメンバを制限した Model Object コレクション属性の値を表します。参照目的に使用することができます。 ClassIds には、受け入れ可能なクラス識別子(オブジェクト タイプなど)のリストが含まれます。このリストが空でない場合、コレクションには、クラス識別子がリストにあるオブジェクトのみが含まれます。リストが空であるか、または NULL ポインタを返す場合、すべてのオブジェクトが含まれます。 オブジェクト クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

メソッド	説明
BSTR * ClassNames()	ClassIds と似ていますが、クラス名（オブジェクト タイプ名など）の SAFEARRAY を返す点が異なります。 オブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ISCModelObjectCollection * Collect(VARIANT Root, VARIANT ClassId [省略可], VARIANT Depth [省略可], VARIANT MustBeOn [省略可], VARIANT MustBeOff [省略可])	それ自身のサブコレクションを表す、Model Objects コレクションを作成します。Collect の呼び出しで指定されたフィルタ条件は、コレクションのメンバに適用されます。 このメソッドは、コレクションが空の場合でも有効なコレクションを作成します。 すべての列挙体は深さ優先です。
long Count()	コレクション内のオブジェクト数。この数にはルート オブジェクトが含まれていません。
long Depth()	コレクション内での繰り返し時の深さ制限。-1 を指定すると、深さの制限がないことを表します。
ISCModelObject * Item(VARIANT nIndex, VARIANT Class [省略可])	Index パラメータによって指定された Model Object コンポーネントの IUnknown ポインタを返します。
SC_ModelObjectFlags MustBeOff()	コレクション内のモデル オブジェクト フラグのフィルタ。
SC_ModelObjectFlags MustBeOn()	コレクション内のモデル オブジェクト フラグのフィルタ。
VARIANT_BOOL Remove(VARIANT Object)	指定されたモデル オブジェクトをモデルから削除します。
ISCModelObject * Root()	コレクション内のルート オブジェクトへのポインタを返します。

ISCModelObjectCollection::Add の引数

Add 関数のシグネチャを示します:

```
ISCModelObject * Add(VARIANT Class,  VARIANT ObjectId)
```

次の表に、Add 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Class	VT_BSTR – クラス名	指定されたクラス名でオブジェクト クラスを特定します。有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
Class	VT_BSTR – オブジェクト タイプのクラス ID	指定された識別子でオブジェクト クラスを特定します。有効なオブジェクト クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ObjectId [省略可]	Empty	API はオブジェクト識別子を新規オブジェクトに割り当てます。

パラメータ	有効なタイプ/値	説明
ObjectId [省略可]	VT_BSTR – 新規オブジェクトのオブジェクト ID	API は指定されたオブジェクト識別子を新規オブジェクトに割り当てます。

ISCModelObjectCollection::Collect の引数

Collect 関数のシグネチャを示します:

```
ISCModelObjectCollection * Collect(VARIANT Root, VARIANT ClassId, VARIANT Depth,
VARIANT MustBeOn, VARIANT MustBeOff)
```

次の表に、Collect 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Root	VT_UNKNOWN – ルート オブジェクトの ISCModelObject ポインタ	コレクションのコンテキスト(親)オブジェクトを指定します。
Root	VT_BSTR – ルート オブジェクトの ID	コレクションのコンテキスト(親)オブジェクトを指定します。
ClassId [省略可]	VT_ARRAY|VT_BSTR – クラス ID の SAFEARRAY	受け入れ可能なクラス識別子の一覧が含まれます。 有効なオブジェクト ID の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId [省略可]	VT_ARRAY|VT_BSTR – クラス名の SAFEARRAY	受け入れ可能なクラス名の一覧が含まれます。 有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId [省略可]	VT_BSTR – クラス ID	単一タイプ コレクションのクラス識別子を指定します。 有効なオブジェクト識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId [省略可]	VT_BSTR – セミコロンで区切られたクラス ID の一覧	受け入れ可能なクラス識別子の一覧が含まれます。 有効なオブジェクト識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId [省略可]	VT_BSTR – クラス名	単一タイプ コレクションのタイプ名を指定します。 有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

パラメータ	有効なタイプ/値	説明
ClassId [省略可]	VT_BSTR – セミコロンで区切られたクラス名の一覧	受け入れ可能なクラス名の一覧が含まれます。 有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId [省略可]	Empty	クラス タイプに関係なく、すべての子を返します。
Depth [省略可]	VT_I4 – 子の最大の深さ。深さ 1 はルート直下の子を返します。−1 の深さ(デフォルト値)は、深さの制限がないことを表します。	指定された深さまでのルートの子を返します。
Depth [省略可]	Empty	ルートのすべての子を返します(深さの制限なし)。
MustBeOn [省略可]	VT_I4 – 設定が必須の SC_ModelObjectFlags	必須フラグのセットを指定します。 SC_ModelObjectFlags の詳細については、この付録の「列挙体」セクションを参照してください。
MustBeOn [省略可]	Empty	SCD_MOF_DONT_CARE がデフォルトになります。
MustBeOff [省略可]	VT_I4 – 設定してはいけない SC_ModelObjectFlags	設定してはいけないフラグ セットを指定します。 SC_ModelObjectFlags の詳細については、この付録の「列挙体」セクションを参照してください。
MustBeOff [省略可]	Empty	SCD_MOF_DONT_CARE がデフォルトになります。

ISCModelObjectCollection::Item の引数

Item 関数のシグネチャを示します:

```
ISCModelObject * Item(VARIANT nIndex,  VARIANT Class)
```

次の表に、Item 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
nIndex	VT_UNKNOWN – ISCModelObject インターフェイスへのポインタ	Model Object ポインタを持つオブジェクトを特定します。
nIndex	VT_BSTR – オブジェクト ID	指定されたオブジェクト識別子を持つオブジェクトを特定します。

パラメータ	有効なタイプ/値	説明
nIndex	VT_BSTR – オブジェクト名	オブジェクト名を使用する場合、Class パラメータも使用する必要があります。指定された名前とオブジェクト クラスを持つオブジェクトを特定します。
Class [省略可]	Empty	nIndex がオブジェクト名でない場合のみ。
Class [省略可]	VT_BSTR – クラス名	nIndex パラメータがオブジェクト名の場合に使用する必要があります。オブジェクトのクラス名を特定します。 有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
Class [省略可]	VT_BSTR – オブジェクト タイプのクラス ID	nIndex パラメータがオブジェクト名の場合に使用する必要があります。オブジェクトのクラス識別子を特定します。 有効なオブジェクト クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCModelObjectCollection::Remove の引数

Remove 関数のシグネチャを示します:

```
VARIANT_BOOL Remove(VARIANT Object)
```

次の表に、Remove 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Object	VT_UNKNOWN。ISCModelObject – オブジェクトへのポインタ	Model Object ポインタによって、削除するオブジェクトを指定します。
Object	VT_BSTR – オブジェクト ID	オブジェクトの識別子によって、削除するオブジェクトを指定します。

ISCModelProperty

ISCModelProperty インターフェイスは、指定されたオブジェクトのプロパティを表します。

次の表に、ISCModelProperty インターフェイスのメソッドを示します。

メソッド	説明
BSTR ClassName()	プロパティのクラス名を返します。 プロパティのクラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

メソッド	説明
BSTR FormatAsString()	プロパティ値を文字列型に変換します。
ISCPropertyValueCollection * PropertyValues()	モデル プロパティ値のコレクションを返します。
long Count()	プロパティ内の値の数が含まれます。
SC_CLSID ClassId()	プロパティのクラス識別子を返します。 プロパティのクラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
SC_ModelPropertyFlags Flags()	プロパティのフラグを返します。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
SC_ValueTypes DataType(VARIANT ValueId [省略可])	指定されたプロパティ値について、ネイティブ値タイプの識別子を返します。 SC_ValueTypes の詳細については、この付録の「列挙体」セクションを参照してください。
VARIANT_BOOL GetValueFacetIds(Long* FacetsTrueBasket, Long* FacetsFalseBasket)	利用可能なプロパティ ファセット ID を取得します。 FacetsTrueBasket は、ファセット ID 番号の SAFEARRAY です。リストされたファセットの値は TRUE になります。 FacetsFalseBasket は、ファセット ID 番号の SAFEARRAY です。リストされたファセットの値は FALSE になります。 プロパティが値を持たない場合、このメソッドは FALSE を返します。
VARIANT_BOOL GetValueFacetNames(BSTR* FacetsTrueBasket,BSTR* FacetsFalseBasket)	利用可能なプロパティ ファセット名を取得します。 FacetsTrueBasket は、ファセット名文字列の SAFEARRAY です。リストされたファセットの値は TRUE になります。 FacetsFalseBasket は、ファセット名文字列の SAFEARRAY です。リストされたファセットの値は FALSE になります。 プロパティが値を持たない場合、このメソッドは FALSE を返します。
VARIANT_BOOL IsValid()	自身が有効な場合、TRUE を返します。
VARIANT_BOOL RemoveAllValues()	プロパティからすべての値を削除します。
VARIANT_BOOL RemoveValue(VARIANT ValueId [省略可])	指定された値をプロパティから削除します。削除後に値が残っていない場合、プロパティ値は NULL になります。 値が削除されると、TRUE を返します。
VARIANT Value(VARIANT ValueId [省略可], VARIANT ValueType [省略可])	要求された形式で指定されたプロパティ値を取得します。 プロパティのデータタイプの詳細については、この付録の「列挙体」セクションの「SC_ValueTypes」を参照してください。

メソッド	説明
Void SetValueFacets(VARIANT* FacetsTrueBasket, VARIANT* FacetsFalseBasket)	プロパティ ファセットに新しい値を割り当てます。 FacetsTrueBasket は TRUE に設定されるファセット一覧です。ファセット ID 番号の SAFEARRAY、ファセット名文字列の SAFEARRAY、またはセミコロンで区切られたファセット名文字列のいずれかとなります。 FacetsFalseBasket は FALSE に設定されるファセット一覧です。ファセット ID 番号の SAFEARRAY、ファセット名文字列の SAFEARRAY、またはセミコロンで区切られたファセット名文字列のいずれかとなります。 プロパティが値を持たない場合、このメソッドは FALSE を返します。
void Value(VARIANT ValueId [省略可], VARIANT ValueType [省略可], VARIANT Val)	指定された値を持つプロパティ値を設定します。

ISCModelProperty::DataType の引数

DataType 関数のシグネチャを示します:

```
SC_ValueTypes DataType(VARIANT ValueId)
```

次の表に、DataType 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ValueId	Empty	プロパティがスカラであるか、または複数値プロパティの全要素が同じデータタイプである場合に有効となります。
ValueId	VT_I4 – インデックス	プロパティがスカラの場合は無視されます。ゼロベース インデックスを使用して、複数値プロパティの要素を指定します。
ValueId	VT_BSTR – 非スカラ要素の名前	プロパティがスカラの場合は無視されます。プロパティが複数値を持つ場合は、名前によって要素を指定します。

ISCModelProperty::RemoveValue の引数

RemoveValue 関数のシグネチャを示します:

```
VARIANT_BOOL RemoveValue(VARIANT ValueId)
```

次の表に、RemoveValue 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ValueId	Empty	スカラ プロパティのみで有効です。

パラメータ	有効なタイプ/値	説明
ValueId	VT_I4 – インデックス	プロパティがスカラの場合は無視されます。ゼロベース インデックスを使用して、複数値プロパティの要素を指定します。
ValueId	VT_BSTR – 非スカラ要素の名前	プロパティがスカラの場合は無視されます。プロパティが複数値を持つ場合は、名前によって要素を指定します。

ISCModelProperty::Value の引数(Get 関数)

Value(Get)関数のシグネチャを示します:

```
VARIANT Value(VARIANT ValueId, VARIANT ValueType)
```

次の表に、Value(Get)関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ValueId [省略可]	Empty	スカラ プロパティのみで有効です。
ValueId [省略可]	VT_BSTR – 非スカラ要素の名前	プロパティがスカラの場合は無視されます。プロパティが複数値を持つ場合は、名前によって要素を指定します。
ValueId [省略可]	VT_I4 – 非スカラ要素のインデックス	プロパティがスカラの場合は無視されます。プロパティが複数値を持つ場合は、ゼロベース インデックスによって要素を指定します。
ValueType [省略可]	Empty	戻り値にネイティブのデータタイプを指定します。
ValueType [省略可]	VT_I4 – SCVT_DEFAULT	戻り値にネイティブのデータタイプを指定します。
ValueType [省略可]	VT_I4 – SCVT_BSTR	戻り値を文字列に変換します。

ISCModelProperty::Value の引数(Set 関数)

Value(Set)関数のシグネチャを示します:

```
void Value(VARIANT ValueId, VARIANT ValueType, VARIANT Val)
```

次の表に、Value(Set)関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ValueId [省略可]	Empty	スカラ プロパティのみで有効です。

パラメータ	有効なタイプ/値	説明
ValueId [省略可]	VT_I4 – 非スカラ プロパティのインデックス	ゼロベース インデックスによって、非スカラ プロパティの値の位置を指定します。 値を-1 にすると、ベクトルの最後に新しい値が追加されます。
ValueId [省略可]	VT_BSTR – 複数値プロパティの要素名	名前によって値の位置を指定します。
ValueType [省略可]	Empty	未使用
Val	プロパティ タイプに依存します。	

ISCModelProperty::GetValueFacetIds の引数

GetValueFacetIds 関数のシグネチャを示します:

```
VARIANT_BOOL GetValueFacetIds(Long* FacetsTrueBasket, Long* FacetsFalseBasket)
```

次の表に、GetValueFacetIds 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
FacetsTrueBasket	SAFEARRAY(VT_I4) – ファセット ID の配列	値が TRUE に設定されたファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsFalseBasket	SAFEARRAY(VT_I4) – ファセット ID の配列	値が FALSE に設定されたファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCModelProperty::GetValueFacetNames の引数

GetValueFacetNames 関数のシグネチャを示します:

```
VARIANT_BOOL GetValueFacetNames(BSTR* FacetsTrueBasket, BSTR*
FacetsFalseBasket)
```

次の表に、GetValueFacetNames 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
FacetsTrueBasket	SAFEARRAY(VT_BSTR) – ファセット名の配列	値が TRUE に設定されたファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsFalseBasket	SAFEARRAY(VT_BSTR) – ファセット名の配列	値が FALSE に設定されたファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCModelProperty::SetValueFacets の引数

SetValueFacets 関数のシグネチャを示します:

```
void SetValueFacets(VARIANT FacetsTrueBasket, VARIANT FacetsFalseBasket)
```

次の表に、SetValueFacets 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
FacetsTrueBasket	SAFEARRAY(VT_I4) – ファセット ID の配列	TRUE に設定されるファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsTrueBasket	SAFEARRAY(VT_BSTR) – ファセット名の配列	TRUE に設定されるファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsTrueBasket	VT_BSTR – セミコロンで区切られたファセット名文字列	TRUE に設定されるファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

パラメータ	有効なタイプ/値	説明
FacetsFalseBasket	SAFEARRAY(VT_I4) – ファセット ID の配列	FALSE に設定されるファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsFalseBasket	SAFEARRAY(VT_BSTR) – ファセット名の配列	FALSE に設定されるファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsFalseBasket	VT_BSTR – セミコロンで区切られたファセット名文字列	FALSE に設定されるファセットの一覧です。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCModelPropertyCollection

ISCModelPropertyCollection インターフェイスは、指定されたモデル オブジェクトのプロパティ コレクションです。フィルタ条件を設定して、このコレクションのメンバを制限することができます。

次の表に、ISCModelPropertyCollection インターフェイスのメソッドを示します。

メソッド	説明
IUnknown _NewEnum()	コレクション列挙子オブジェクトのインスタンスを作成します。
ISCModelProperty * Add(VARIANT ClassId)	バインドされたモデル オブジェクトの新規プロパティを、存在しない場合に作成します。
SC_CLSID * ClassIds()	プロパティ コレクション内のプロパティ クラス識別子の SAFEARRAY を返します。 このコレクションの作成時にメンバを制限した ModelProperties コレクション属性の値を表します。参照目的に使用することができます。 ClassIds には、受け入れ可能なクラス識別子（プロパティ クラスなど）の配列が含まれます。この一覧が空でない場合、プロパティ コレクションには、クラス識別子が一覧に存在するプロパティのみが含まれます。一覧が空であるか、または呼び出し元が NULL ポインタを指定する場合、コレクションには、オブジェクトが所有するすべてのプロパティが含まれます。 プロパティのクラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

メソッド	説明
BSTR * ClassNames()	ClassIdsプロパティと同様です。ただし、プロパティコレクション内のプロパティ タイプ名の SAFEARRAY を返します。 プロパティのクラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
long Count()	コレクション内のプロパティ数
VARIANT_BOOL HasProperty(VARIANT ClassId, VARIANT MustBeOn [省略可], VARIANT MustBeOff [省略可])	オブジェクトが、受け渡されたクラスのプロパティを所有する場合、TRUE を返します。 プロパティが、コレクションの ClassIds、MustBeOn、および MustBeOff 属性を満たさない場合、それらのプロパティは存在しないものとして処理されます。 任意指定のパラメータを使用して、代わりの MustBeOn および MustBeOff を指定することもできます。
VARIANT_BOOL HasPropertyFacets(VARIANT ClassId, VARIANT MustBeOn [省略可], VARIANT MustBeOff [省略可], VARIANT FacetsMustBeSet [省略可])	オブジェクトが、受け渡されたクラスのプロパティを所有する場合、TRUE を返します。 プロパティが、コレクションの ClassIds、MustBeOn、および MustBeOff 属性を満たさない場合、それらのプロパティは存在しないものとして処理されます。 任意指定のパラメータを使用して、FlagsMustBeOn、FlagsMustBeOff、および FacetsMustBeSet を指定することもできます。 FacetsMustBeSet は、プロパティが 1 つ以上のファセットを持つ必要があることを示します。パラメータは、ファセット ID 番号の SAFEARRAY、ファセット名文字列の SAFEARRAY、またはセミコロンで区切られたファセット名文字列のいずれかとなります。
ISCModelProperty * Item(VARIANT Class)	モデル オブジェクトのプロパティを返します。 このメソッドはプロパティが存在するかどうかを確認します。プロパティが存在しない場合、メソッドはプロパティの記述を作成し、ISCModelProperty インスタンスを返してから、そのプロパティに NULL フラグを設定します。新しいプロパティ値を設定するには、インスタンスの Value プロパティを使用します。ただし、設定前に値を取得することはできません。 このメソッドを使用して、ReadOnly や Maintained By the Toolなどのプロパティに対して、ISCModelProperty のインスタンスを作成することができます。これらのプロパティの値を変更したり指定することはできません。ただし、プロパティ フラグやデータタイプなどは、コレクションがプロパティ インスタンスを持たない場合でも利用できます。HasProperty を使用して、モデル オブジェクト インスタンスにプロパティが存在するかを確認できます。

メソッド	説明
SC_ModelPropertyFlags MustBeOff()	コレクション内のプロパティ フラグのフィルタ。このフィルタは、プロパティ コレクションが ISCModelObject::CollectProperties メソッドによって作成されたときに設定されます。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
SC_ModelPropertyFlags MustBeOn()	コレクション内のプロパティ フラグのフィルタ。このフィルタは、プロパティ コレクションが ISCModelObject::CollectProperties メソッドによって作成されたときに設定されます。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
VARIANT_BOOL Remove(VARIANT ClassId)	指定されたプロパティをバインドされたオブジェクトから削除します。 呼び出し処理が成功すると、削除されたプロパティとのすべてのバインドが無効になります。クライアントは、そのような関連付けを表していたすべての ISCModelProperty ポインタ、および関連するすべての Value Collection および Value ポインタを開放する必要があります。インターフェイスへの呼び出しは失敗し、IsValid メソッドは FALSE を返します。

ISCModelPropertyCollection::Add の引数

Add 関数のシグネチャを示します:

```
ISCModelProperty * Add(VARIANT ClassId)
```

次の表に、Add 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ClassId	VT_BSTR – プロパティ クラス名	新しいプロパティ タイプ名を指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId	VT_BSTR – プロパティ ID	新しいプロパティ クラス識別子を指定します。 有効なプロパティ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCModelPropertyCollection::HasProperty の引数

HasProperty 関数のシグネチャを示します:

```
VARIANT_BOOL HasProperty(VARIANT ClassId, VARIANT MustBeOn, VARIANT MustBeOff)
```

次の表に、HasProperty 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ClassId	VT_BSTR – プロパティ名	プロパティのクラス名を指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId	VT_BSTR – プロパティ ID	プロパティのクラス識別子を指定します。 有効なプロパティ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
MustBeOn [省略可]	VT_I4 – 設定が必須の SC_ModelPropertyFlags	必須フラグのセットを指定します。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
MustBeOn [省略可]	Empty	プロパティ コレクションの作成に使用された MustBeOn フィルタが、デフォルトに設定されます。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
MustBeOff [省略可]	VT_I4 – オフにする必要がある SC_ModelPropertyFlags	オフにする必要があるフラグ セットを指定します。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
MustBeOff [省略可]	Empty	プロパティ コレクションの作成に使用された MustBeOff フィルタが、デフォルトに設定されます。

ISCModelPropertyCollection::HasPropertyFacets の引数

HasPropertyFacets 関数のシグネチャを示します:

```
VARIANT_BOOL HasPropertyFacets(VARIANT ClassId, VARIANT FlagsMustBeOn, VARIANT FlagsMustBeOff, VARIANT FacetsMustBeSet)
```

次の表に、HasPropertyFacets 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ClassId	VT_BSTR – プロパティ名	プロパティのクラス名を指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId	VT_BSTR – プロパティ ID	プロパティのクラス識別子を指定します。 有効なプロパティ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
FlagsMustBeOn [省略可]	VT_I4 – 設定が必須の SC_ModelPropertyFlags	必須フラグのセットを指定します。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
FlagsMustBeOn [省略可]	Empty	プロパティ コレクションの作成に使用された MustBeOn フィルタが、デフォルトに設定されます。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
FlagsMustBeOff [省略可]	VT_I4 – オフにする必要がある SC_ModelPropertyFlags	オフにする必要があるフラグ セットを指定します。 SC_ModelPropertyFlags の詳細については、この付録の「列挙体」セクションを参照してください。
FlagsMustBeOff [省略可]	Empty	プロパティ コレクションの作成に使用された MustBeOff フィルタが、デフォルトに設定されます。
FacetsMustBeSet [省略可]	SAFEARRAY(VT_I4) – ファセット ID の配列	プロパティが 1 つ以上のファセットを持つ必要があることを示します。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

パラメータ	有効なタイプ/値	説明
FacetsMustBeSet [省略可]	SAFEARRAY(VT_BSTR) – ファセット名の配列	プロパティが 1 つ以上のファセットを持つ必要があることを示します。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsMustBeSet [省略可]	VT_BSTR – セミコロンで区切られたファセット名文字列	プロパティが 1 つ以上のファセットを持つ必要があることを示します。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
FacetsMustBeSet [省略可]	Empty	ファセットの要件はありません。

ISCModelPropertyCollection::Item の引数

Item 関数のシグネチャを示します:

```
ISCModelProperty * Item(VARIANT Class)
```

次の表に、Item 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Class	VT_BSTR – プロパティ ID	プロパティ クラス識別子を指定します。 有効なプロパティ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
Class	VT_BSTR – プロパティ名	プロパティ クラス名を指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCModelPropertyCollection::Remove の引数

Remove 関数のシグネチャを示します:

```
VARIANT_BOOL Remove(VARIANT ClassId)
```

次の表に、Remove 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ClassId	ISCModelProperty *	Model Property オブジェクトでプロパティを指定します。

パラメータ	有効なタイプ/値	説明
ClassId	VT_BSTR – プロパティ名	クラス名でプロパティを指定します。 有効なプロパティ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
ClassId	VT_BSTR – プロパティ ID	クラス識別子でプロパティを指定します。 有効なプロパティ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCModelSet

Model Setコンポーネントを使用して、モデル セットで階層的に構成されたコレクションのメンバにアクセスすることができます。

次の表に、ISCModelSet インターフェイスのメソッドを示します。

メソッド	説明
SC_MODELTYPEID ClassId()	モデル セットに関連付けられたメタデータのクラス識別子。 メタデータ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
BSTR ClassName()	モデル セットに関連付けられたメタデータのクラス名。 メタデータ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
VARIANT_BOOL DirtyBit()	モデル セットのデータが変更されたことを示すフラグを返します。
void DirtyBit(VARIANT_BOOL)	モデル セットのデータが変更されたことを示すフラグを設定します。
SC_MODELTYPEID ModelSetId()	モデル セットの識別子を返します。
BSTR Name()	永続ユニットの名前を返します。
ISCModelSet * Owner()	所有者のモデル セットへのポインタ。永続ユニットのトップ モデル セットには NULL を返します。
ISCModelSetCollection * OwnedModelSets()	直接所有するモデル セットのコレクションを提供します。
SC_MODELTYPEID PersistenceUnitId()	モデル セットが含まれている永続ユニットの識別子。
ISCPropertyBag* PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可])	モデル セットのプロパティを含むプロパティ バッグを返します。 モデル セットのプロパティがプロパティ バッグに含まれるのは、値を持つ場合のみです。プロパティに値セットがない場合、プロパティ バッグにはリストされません。

メソッド	説明
void PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可], ISCPropertyBag * propBag)	モデル セットに、指定されたプロパティ バッグのプロパティを設定します。

ISCModelSet::PropertyBag の引数（Get 関数）

PropertyBag（Get）関数のシグネチャを示します:

```
ISCPropertyBag * PropertyBag(VARIANT List, VARIANT AsString)
```

次の表に、PropertyBag（Get）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]	VT_BSTR – セミコロンで区切られたプロパティー覧	モデル セット プロパティの一覧を指定します。一覧に指定されたプロパティのみが、返されるプロパティ バッグ内に配置されます。 プロパティ名の情報については、この付録の「永続ユニットおよび永続ユニット コレクションのプロパティ バッグ」セクションを参照してください。
List [省略可]	Empty	すべてのプロパティ セットを要求します。
AsString [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、プロパティ バッグのすべての値が文字列型になります。デフォルトは FALSE で、すべての値はそれぞれのネイティブ型になります。
AsString [省略可]	Empty	プロパティ バッグのすべての値はネイティブ型になります。

ISCModelSet::PropertyBag の引数（Set 関数）

PropertyBag（Set）関数のシグネチャを示します:

```
void PropertyBag(VARIANT List, VARIANT AsString, ISCPropertyBag * propBag)
```

次の表に、PropertyBag（Set）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]		未使用
AsString [省略可]		未使用
propBag	ISCPropertyBag *	処理対象のモデル セット プロパティを含むプロパティ バッグのポインタ。

ISCModelSetCollection

Model Set Collection には、所有者のモデル セットが直接所有するすべてのモデル セットが含まれています。

次の表に、ISCModelSetCollection インターフェイスのメソッドを示します。

メソッド	説明
IUnknown _NewEnum()	モデル セット列挙子オブジェクトのインスタンスを作成します。
long Count()	コレクション内のモデル セット数
ISCPersistenceUnit * Item(VARIANT nIndex)	ModelSet コンポーネントのポインタを返します。
ISCModelSet * Owner()	所有者のモデルセットへのポインタを返します。

ISCModelSetCollection::Item の引数

Item 関数のシグネチャを示します:

```
ISCModelSet * Item(VARIANT nIndex)
```

次の表に、Item 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
nIndex	VT_UNKNOWN – ISCPersistenceUnit へのポインタ	Model Set オブジェクトの複製を作成します。
nIndex	VT_I4 – モデル セット コレクション内のモデル セットのインデックス	コレクション内の順序位置。ゼロベース インデックスです。
nIndex	VT_BSTR – モデル セット ID	モデル セットの識別子。
nIndex	VT_BSTR – メタデータ クラス ID	モデル セットに関連付けられたメタデータのクラス識別子。 メタデータ クラス識別子の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。
nIndex	VT_BSTR – メタデータ クラス名	モデル セットに関連付けられたメタデータのクラス名。 メタデータ クラス名の詳細については、付録 B「AllFusion ERwin DM メタモデル」を参照してください。

ISCPersistenceUnit

Persistence Unit は、アプリケーション内で既存の外部レベル永続ユニットに接続するために必要な情報をカプセル化します。

次の表に、ISCPersistenceUnit インターフェイスのメソッドを示します。

メソッド	説明
VARIANT_BOOL DirtyBit()	永続ユニットのデータが変更されたことを示すフラグを返します。
void DirtyBit(VARIANT_BOOL)	永続ユニットのデータが変更されたことを示すフラグを設定します。
VARIANT_BOOL HasSession()	ユニットに 1 つ以上の接続済みセッションがある場合、TRUE を返します。
VARIANT_BOOL IsValid()	自身が有効な場合、TRUE を返します。
ISCModelSet * ModelSet()	永続ユニットのトップ モデル セットのポインタを返します。
BSTR Name()	永続ユニットの名前を返します。
SC_MODELTYPEID ObjectId()	永続ユニットの識別子を返します。
ISCPropertyBag* PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可])	永続ユニットのプロパティを持つプロパティ バッグを返します。 永続ユニットのプロパティがプロパティ バッグに含まれるのは、値を持つ場合のみです。プロパティに値セットがない場合、プロパティ バッグにはリストされません。 プロパティの詳細な説明については、この付録の「永続ユニットおよび永続ユニット コレクションのプロパティ バッグ」セクションを参照してください。
void PropertyBag(VARIANT List [省略可], VARIANT AsString [省略可], ISCPropertyBag * propBag)	永続ユニットに、指定されたプロパティ バッグのプロパティを設定します。
VARIANT_BOOL Save(VARIANT Locator [省略可], VARIANT Disposition [省略可])	モデル データを外部ストレージに保存します。コミットされていないトランザクションは無視されます。

ISCPersistenceUnit::PropertyBag の引数（Get 関数）

PropertyBag（Get）関数のシグネチャを示します:

```
ISCPropertyBag * PropertyBag(VARIANT List, VARIANT AsString)
```

次の表に、PropertyBag（Get）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]	VT_BSTR – セミコロンで区切られたプロパティー覧	ユニット プロパティの一覧を指定します。一覧に指定されたプロパティのみが、返されるプロパティ バッグ内に配置されます。 有効なプロパティ名の詳細については、この付録の「永続ユニットおよび永続ユニット コレクションのプロパティ バッグ」セクションを参照してください。
List [省略可]	Empty	すべてのプロパティ セットを要求します。
AsString [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、プロパティ バッグのすべての値が文字列型になります。デフォルトは FALSE で、すべての値はそれぞれのネイティブ型になります。
AsString [省略可]	Empty	プロパティ バッグのすべての値はネイティブ型になります。

ISCPersistenceUnit::PropertyBag の引数（Set 関数）

PropertyBag（Set）関数のシグネチャを示します:

```
void PropertyBag(VARIANT List, VARIANT AsString, ISCPropertyBag * propBag)
```

次の表に、PropertyBag（Set）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]		未使用
AsString [省略可]		未使用
propBag	ISCPropertyBag *	処理対象のユニット プロパティを含むプロパティ バッグのポインタ。

ISCPersistenceUnit::Save の引数

Save 関数のシグネチャを示します:

```
VARIANT_BOOL Save(VARIANT Locator, VARIANT Disposition)
```

次の表に、Save 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator [省略可]	VT_BSTR – ストレージの場所を表す完全パス	永続ユニット データ ソースの新しい保存場所です。ファイルまたは AllFusion Model Manager 項目の場所を示す文字列として表されます。ストレージへのアクセスに必要な属性も含めます。 Locator パラメータの形式の詳細については、この付録の「Locator プロパティ」を参照してください。
Locator [省略可]	Empty	永続ユニットの元の場所を使用します。
Disposition [省略可]	VT_BSTR – キーワード パラメータの一覧	読み取り専用など、アクセス属性の変更を指定します。

ISCPersistenceUnitCollection

ISCPersistenceUnitCollection には、アプリケーションに読み込まれたすべての外部レベル永続ユニットが含まれています。アクティブな各データ モデルに対し、1 つのエントリが含まれています。

次の表に、ISCPersistenceUnitCollection インターフェイスのメソッドを示します。

メソッド	説明
IUnknown _NewEnum()	ユニット列挙子オブジェクトのインスタンスを作成します。
ISCPersistenceUnit * Add(VARIANT Locator, VARIANT Disposition [省略可])	新規永続ユニットをユニット コレクションに追加します。
VARIANT_BOOL Clear()	すべてのユニットをコレクションから消去します。
long Count()	コレクション内の永続ユニット数
ISCPersistenceUnit * Create(ISCPropertyBag * PropertyBag, VARIANT ObjectId [省略可])	新規ユニットを作成して、コレクションに登録します。 プロパティ バッグ内のプロパティの詳細については、この付録の「永続ユニットと永続ユニット コレクションのプロパティ バッグ」セクションを参照してください。
ISCPersistenceUnit * Item(VARIANT nIndex)	PersistenceUnit コンポーネントの IUnknown ポインタを返します。
VARIANT_BOOL Remove(VARIANT Selector, VARIANT Save [省略可])	永続ユニットをコレクションから削除します。

ISCPersistenceUnitCollection::Add の引数

Add 関数のシグネチャを示します:

```
ISCPersistenceUnit * Add(VARIANT Locator, VARIANT Disposition)
```

次の表に、Add 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Locator	VT_BSTR – 永続ユニットの場所	永続ユニット データ ソースの場所を指定します。ファイルまたは AllFusion Model Manager 項目の場所を示す文字列として表されます。ストレージへのアクセスに必要な属性も含めます。 Locator パラメータの詳細については、この付録の「Locator プロパティ」セクションを参照してください。
Disposition [省略可]	VT_BSTR – キーワード パラメータの一覧	読み取り専用などのアクセス属性を指定します。 Disposition パラメータの詳細については、この付録の「Disposition プロパティ」セクションを参照してください。

ISCPersistenceUnitCollection::Create の引数

Create 関数のシグネチャを示します:

```
ISCPersistenceUnit * Create(ISCPropertyBag * Property Bag, VARIANT ObjectId)
```

次の表に、Create 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Property Bag	ISCPropertyBag * – Property Bag オブジェクトへのポインタ	モデル タイプなど、作成時の必須および任意のプロパティを指定します。 有効なプロパティ名とフォーマットの詳細については、この付録の「永続ユニットと永続ユニット コレクションのプロパティ バッグ」セクションを参照してください。
ObjectId [省略可]	Empty	新規永続ユニットの ID を生成します。
ObjectId [省略可]	VT_BSTR – 新規永続ユニットのオブジェクト ID	新規永続ユニットの識別子を指定します。

ISCPersistenceUnitCollection::Item の引数

Item 関数のシグネチャを示します:

```
ISCPersistenceUnit * Item(VARIANT nIndex)
```

次の表に、Item 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
nIndex	VT_UNKNOWN – ISCPersistenceUnit へのポインタ	Persistence Unit オブジェクトの複製を作成します。
nIndex	VT_I4 – 永続ユニット コレクション内の永続ユニットのインデックス	コレクション内の順序位置。ゼロベースインデックスです。
nIndex	VT_BSTR – 永続ユニットの ID	アプリケーション レベルで一意な永続ユニットの識別子。

ISCPersistenceUnitCollection::Remove の引数

Remove 関数のシグネチャを示します:

```
VARIANT_BOOL Remove(VARIANT Selector, VARIANT Save)
```

次の表に、Remove 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Selector	VT_UNKNOWN – ISCPersistenceUnit インターフェイスへのポインタ	永続ユニットを指定します。
Selector	VT_BSTR – 永続ユニットの ID	アプリケーション レベルで一意な永続ユニットの識別子。
Selector	VT_I4 – 永続ユニット コレクション内の永続ユニットのインデックス	コレクション内の順序位置。ゼロベースインデックスです。
Save [省略可]	VT_BOOL	TRUE に設定すると、コレクションから削除する前に永続ユニットを保存します。デフォルトでは、未保存のすべてのデータが保存されます。ただし、Save パラメータが FALSE であるか、またはユニットが一時的な状態で Locator プロパティが未指定の場合は保存されません。

注: アプリケーションを終了する前に、モデルを閉じる必要があります。ご使用のコードに次の行を追加すると、アプリケーションを終了する前に明示的にモデルを閉じることができます。

...

SaveNewPersistenceUnit(ThePersistenceUnit, DefaultFileName)

TheApplication.PersistenceUnits.Remove(ThePersistenceUnit, False)

...

ISCPropertyBag

ISCPropertyBag インターフェイスを使用して、ISCApplicationEnvironment、ISCPersistenceUnit、および ISCModelSet のプロパティを設定したり、これらのプロパティにアクセスすることができます。また、ISCPropertyBag を使用して、新規永続ユニットのプロパティを設定することもできます。

次の表に、ISCPropertyBag インターフェイスのメソッドを示します。

メソッド	説明
VARIANT_BOOL Add(BSTR Name, VARIANT Value)	新規プロパティをバッグに追加します。名前の重複は確認しません。プロパティがバッグに追加された場合、TRUE を返します。そうでない場合は FALSE を返します。
void ClearAll()	すべてのプロパティをバッグから削除します。
long Count()	プロパティの数を返します。
BSTR Name(long PropertyIdx)	バッグ内の指定されたプロパティ名を取得します。
VARIANT Value(VARIANT Property)	バッグ内の指定されたプロパティを取得します。
void Value(VARIANT Property, VARIANT Val)	バッグ内の指定されたプロパティを設定します。

ISCPropertyBag::Add の引数

Add 関数のシグネチャを示します:

```
VARIANT_BOOL Add(BSTR Name, VARIANT Value)
```

次の表に、Add 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Name	BSTR	新規プロパティの名前。
Value	プロパティに依存します。	新規プロパティの値。

ISCPropertyBag::Name の引数

Name 関数のシグネチャを示します:

```
BSTR Name(long PropertyIdx)
```

次の表に、Name 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
PropertyIdx	Long	要求した名前のゼロベース インデックス。

ISCPropertyBag::Value の引数（Get 関数）

Value（Get）関数のシグネチャを示します:

```
VARIANT Value(VARIANT Property)
```

次の表に、Value（Get）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Property	VT_BSTR – プロパティ名	取得するプロパティを指定します。
Property	VT_I4 – プロパティのインデックス	プロパティ バッグのゼロベース プロパティ インデックス

ISCPropertyBag::Value の引数（Set 関数）

Value（Set）関数のシグネチャを示します:

```
void Value(VARIANT Property, VARIANT Val)
```

次の表に、Value（Set）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Property	VT_BSTR – プロパティ名	更新するプロパティを指定します。
Val	プロパティに依存します。	指定されたプロパティの値。

ISCPropertyValue

ISCPropertyValue インターフェイスは、指定されたプロパティの単一値です。

次の表に、ISCPropertyValue インターフェイスのメソッドを示します。

メソッド	説明
SC_ValueTypes * GetSupportedValueIdTypes()	現在の値の識別子でサポートする値タイプの一覧をグループ化して、SAFEARRAY として返します。 GetValue メソッドは、現在の値を、返された一覧にコードが含まれている任意の値タイプに変換できる必要があります。一覧が空の場合、値はネイティブ形式（デフォルトなど）でのみ利用可能です。参照プロパティは、空の一覧を返す必要があります。
SC_ValueTypes * GetSupportedValueTypes()	サポートする値タイプの一覧をグループ化し、SAFEARRAY として返します。 GetValueId メソッドは、現在の値を、返された一覧にコードが含まれている任意の値タイプに変換できる必要があります。一覧が空の場合、現在の識別子はネイティブ（デフォルトなど）形式でのみ利用可能です。
SC_CLSID PropertyClassId()	現在のプロパティのクラス識別子を返します。
BSTR PropertyClassName()	現在のプロパティのクラス名を返します。
VARIANT Value(VARIANT ValueType [省略可])	現在の値を指定された値タイプに変換します。 値のデータタイプの詳細については、この付録の「列挙体」セクションの「SC_ValueTypes」を参照してください。
VARIANT ValueId(VARIANT ValueType [省略可])	非スカラ プロパティの値を一意に識別します。
SC_ValueTypes ValueIdType()	非スカラ プロパティの値を識別する ValueId のデフォルト タイプを返します。
SC_ValueTypes ValueType()	プロパティ値のデフォルト タイプを返します。

ISCPropertyValue::ValueId の引数

ValueId 関数のシグネチャを示します:

```
VARIANT ValueId(VARIANT ValueType)
```

次の表に、ValueId 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ValueType [省略可]	VT_I4 – SCVT_I2 または SCVT_I4	プロパティがスカラの場合、VT_EMPTY を返します。プロパティが非スカラの場合は、プロパティのゼロベース インデックスの値を返します。

パラメータ	有効なタイプ/値	説明
ValueType [省略可]	VT_I4 – SCVT_BSTR	プロパティがスカラの場合、VT_EMPTY を返します。利用可能な場合、非スカラ プロパティ メンバの名前、またはメンバのインデックスを返します。
ValueType [省略可]	VT_I4 – SCVT_DEFAULT	プロパティがスカラの場合、VT_EMPTY を返します。プロパティが非スカラの場合、プロパティのゼロベース インデックス値を返します。
ValueType [省略可]	Empty	SCVT_DEFAULT がデフォルトになります。

ISCPropertyValue::Value の引数

Value 関数のシグネチャを示します:

```
VARIANT Value(VARIANT ValueType)
```

次の表に、Value 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ValueType [省略可]	VT_I4 – SCVT_DEFAULT	ネイティブ形式のプロパティ値を要求します。
ValueType [省略可]	VT_I4 – SCVT_BSTR	プロパティ値の文字列変換を要求します。
ValueType [省略可]	VT_I4 – プロパティ タイプ	タイプ変換の対象を指定します。
ValueType [省略可]	Empty	SCVT_DEFAULT がデフォルトになります。

ISCPropertyValueCollection

ISCPropertyValueCollection インターフェイスは、非スカラ プロパティ値のコレクションです。

次の表に、ISCPropertyValueCollection インターフェイスのメソッドを示します。

メソッド	説明
IUnknown _NewEnum()	コレクション列挙子オブジェクトのインスタンスを作成します。
long Count()	コレクション内の値の数。
ISCPropertyValue * Item(VARIANT ValueId)	プロパティ値コレクションから 1 つの値を返します。

メソッド	説明
VARIANT_BOOL Facet (VARIANT Facet)	ファセットを取得します。ファセットが未設定の場合は失敗します。 ファセットは、ファセット ID またはファセット名のいずれかで指定します。
void Facet (VARIANT Facet, VARIANT_BOOL Val)	指定された値をファセットに設定します。 ファセットは、ファセット ID またはファセット名のいずれかで指定します。
VARIANT_BOOL RemoveFacet (VARIANT Facet)	ファセットを削除して未設定の状態にします。 ファセットは、ファセット ID またはファセット名のいずれかで指定します。

ISCPropertyValueCollection::Item の引数

Item 関数のシグネチャを示します:

```
ISCPropertyValue * Item(VARIANT ValueId)
```

次の表に、Item 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
ValueId	VT_I4 – 複数値プロパティの要素インデックス	ゼロベース インデックスを使用して要素を指定します。
ValueId	VT_BSTR – 複数値プロパティの要素名	名前を使用して要素を指定します。

ISCPropertyValueCollection::Facet の引数（Get 関数）

Facet（Get）関数のシグネチャを示します:

```
VARIANT_BOOL Facet (VARIANT Facet)
```

次の表に、Facet（Get）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Facet	VT_I4 – ファセット ID	ファセット値を取得します。ファセットが未設定の場合は失敗します。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

パラメータ	有効なタイプ/値	説明
Facet	VT_BSTR – ファセット名	ファセット値を取得します。ファセットが未設定の場合は失敗します。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCPropertyValueCollection::Facet の引数（Set 関数）

Facet（Set）関数のシグネチャを示します:

```
Void Facet (VARIANT Facet, VARIANT_BOOL Val)
```

次の表に、Facet（Set）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Facet	VT_I4 – ファセット ID	指定された値をファセットに設定します。ファセットが未設定の場合は失敗します。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。
Facet	VT_BSTR – ファセット名	指定された値をファセットに設定します。ファセットが未設定の場合は失敗します。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCPropertyValueCollection::RemoveFacet の引数

RemoveFacet 関数のシグネチャを示します:

```
VARIANT_BOOL RemoveFacet (VARIANT Facet)
```

次の表に、RemoveFacet 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Facet	VT_I4 – ファセット ID	ファセットを削除して未設定の状態にします。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

パラメータ	有効なタイプ/値	説明
Facet	VT_BSTR – ファセット名	ファセットを削除して未設定の状態にします。 詳細については、この付録の「アプリケーション環境のプロパティ バッグ」セクションを参照してください。

ISCSession

ISCSession インターフェイスは、API クライアントとモデルの間のアクティブな接続です。

次の表に、ISCSession インターフェイスのメソッドを示します。

メソッド	説明
VARIANT BeginTransaction()	セッションにトランザクションを開きます。メソッドはトランザクションの識別子を返します。この識別子を使用して、コミットおよびロールバック操作をスコープします。アプリケーションがネストしたトランザクションをサポートしない場合、VT_EMPTY を返します。 トランザクションのネストは暗黙的に実行されます。API クライアントが BeginTransaction を呼び出したときに、トランザクションが既に開いている場合、既存のトランザクションの内部に新規トランザクションがネストされます。
VARIANT BeginNamedTransaction(BSTR Name, VARIANT PropertyBag [省略可])	セッションにトランザクションを開きます。BeginTransaction に似ていますが、トランザクション名と追加のプロパティを指定できます。 トランザクションのプロパティについては、この付録の「セッションのプロパティ バッグ」セクションを参照してください。
VARIANT_BOOL ChangeAccess(SC_SessionFlags Flags)	モデル アクセスを指定されたレベルに変更します。 有効な SC_SessionFlags については、この付録の「列挙体」セクションを参照してください。
VARIANT_BOOL Close()	関連付けられた永続ユニットまたはモデル セットから自身の接続を解除します。
VARIANT_BOOL CommitTransaction(VARIANT TransactionId)	指定されたトランザクションと、それに含まれるすべてのネストしたトランザクションをコミットします。
SC_SessionFlags Flags()	セッションに関連付けられたフラグ セットを返します。 SC_SessionFlags の詳細については、この付録の「列挙体」セクションを参照してください。
VARIANT_BOOL IsValid()	自身が有効な場合、TRUE を返します。

メソッド	説明
VARIANT_BOOL IsTransactionEmpty(VARIANT All [省略可])	外部トランザクションを開始してから、または現在のトランザクション中にデータの変更が適用されていない場合、TRUE を返します。 開いているトランザクションが存在しない場合も TRUE を返します。
SC_SessionLevel Level()	永続ユニットまたはモデルがバインドされたレベルを返します。 この値が有効なのは、セッションが開いている場合のみです。 SC_SessionLevel の詳細については、この付録の「列挙体」セクションを参照してください。
VARIANT_BOOL IsOpen()	セッションが開いている場合のみ、TRUE を返します。
ISCModelObjectCollection * ModelObjects()	セッションの ModelObject コレクションを作成します。 作成されたコレクションには、永続ユニットまたはモデル セットに関連付けられたすべてのオブジェクトが含まれます。
SC_MODELTYPEID ModelSetId()	セッションに関連付けられたモデル セットの識別子を返します。
BSTR Name()	関連付けられた永続ユニットまたはモデル セットの名前です。 有効な名前が含まれるのは、セッションが Opened 状態の場合のみです。
VARIANT_BOOL Open(IUnknown * Target, VARIANT Level [省略可], VARIANT Flags [省略可])	Target パラメータで指定された永続ユニット、モデル セット、または組み込みメタモデルにバインドします。
ISCPersistenceUnit * PersistenceUnit()	セッションに関連付けられた永続ユニット。有効なポインタが含まれるのは、セッションが Opened 状態の場合のみです。
long TransactionDepth()	ネストしたトランザクションの現在の深さレベルを返します。アクティブなトランザクションが存在しない場合は、ゼロを返します。

ISCSession::BeginNamedTransaction の引数

BeginNamedTransaction 関数のシグネチャを示します:

```
VARIANT_BOOL BeginNamedTransaction(BSTR Name, VARIANT PropertyBag )
```

次の表に、BeginNamedTransaction 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Name	BSTR	新規トランザクションの名前を指定します。
PropertyBag	Empty	指定可能なパラメータなし

パラメータ	有効なタイプ/値	説明
PropertyBag	VT_UNKNOWN – Property Bag オブジェクトへのポインタ	トランザクション プロパティのコレクション トランザクションのプロパティについては、この付録の「セッションのプロパティ バッグ」セクションを参照してください。

ISCSession::CommitTransaction の引数

CommitTransaction 関数のシグネチャを示します:

```
VARIANT_BOOL CommitTransaction(VARIANT TransactionId)
```

次の表に、CommitTransaction 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
TransactionId	セッションの ID	トランザクションの識別子を指定します。

ISCSession::IsTransactionEmpty の引数

IsTransactionEmpty 関数のシグネチャを示します:

```
VARIANT_BOOL IsTransactionEmpty(VARIANT All)
```

次の表に、IsTransactionEmpty 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
All	Empty	現在のトランザクションの状態を要求します。
All	VT_BOOL, FALSE	現在のトランザクションの状態を要求します。
All	VT_BOOL, TRUE	外部トランザクションが開始してからのすべてのトランザクションの状態を要求します。

ISCSession::Open の引数

Open 関数のシグネチャを示します:

```
VARIANT_BOOL Open(ISCPersistenceUnit * Unit, VARIANT Level, VARIANT Flags)
```

次の表に、Open 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Target	ISCPersistenceUnit * – 永続ユニットへのポインタ	アタッチする永続ユニットを指定します。
Target	ISCModelSet * – モデル セットへのポインタ	アタッチするモデル セットを指定します。
Target	ISCPropertyBag * – プロパティ バッグへのポインタ	アタッチする組み込みメタモデルの記述を含むプロパティ バッグを指定します。
Level [省略可]	Empty	SCD_SL_M0 がデフォルトになります。
Level [省略可]	SCD_SL_M0	データ レベル アクセス。
Level [省略可]	SCD_SL_M1	メタモデル アクセス。
Flags [省略可]	Empty	SCD_SF_NONE がデフォルトになります。
Flags [省略可]	SCD_SF_NONE	他のセッションは、アタッチされた永続ユニットにアクセスできます。
Flags [省略可]	SCD_SF_EXCLUSIVE	他のセッションは、アタッチされた永続ユニットにアクセスできません。

ISCSession::RollbackTransaction の引数

RollbackTransaction 関数のシグネチャを示します:

```
VARIANT_BOOL RollbackTransaction(VARIANT TransactionId)
```

次の表に、RollbackTransaction 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
TransactionId	セッションの ID	トランザクションの識別子を指定します。

ISCSessionCollection

Session Collection には、API クライアントとアプリケーションの間のアクティブな接続が含まれます。

次の表に、ISCSessionCollection インターフェイスのメソッドを示します。

メソッド	説明
IUnknown _NewEnum()	セッション列挙子オブジェクトのインスタンスを作成します。
ISCSession * Add()	新しい Session オブジェクトを作成し、コレクションに追加します。
VARIANT_BOOL Clear()	すべての Session オブジェクトをコレクションから削除します。
long Count()	コレクション内のセッションの数。
ISCSession * Item(long nIndex)	順序位置によって指定されたセッションを返します。
VARIANT_BOOL Remove(VARIANT SessionId)	Session オブジェクトをコレクションから削除します。開いているセッションは、削除前に閉じられます。コミットされたすべての変更は永続ユニットに保存されます。

ISCSessionCollection::Item の引数

Item 関数のシグネチャを示します:

```
ISCSession * Item(long Index)
```

次の表に、Item 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Index	long-Index	セッションのゼロベース インデックスを指定します。

ISCSessionCollection::Remove の引数

Remove 関数のシグネチャを示します:

```
VARIANT_BOOL Remove(VARIANT SessionId)
```

次の表に、Remove 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
SessionId	VT_UNKNOWN – ISCSession インターフェイスへのポインタ	Session オブジェクトを使用してセッションを指定します。
SessionId	VT_I4 – セッション コレクション内のインデックス	セッションのゼロベース インデックスを指定します。

列挙体

このセクションでは、API に含まれるさまざまな列挙体について説明します。列挙体は、さまざまなプロパティに対する有効値を定義します。

SC_ModelDirectoryFlags

次の表に、SC_ModelDirectoryFlags のプロパティと列挙値を示します。

プロパティ	列挙値	説明
SCD_MDF_DIRECTORY	0	ディレクトリ
SCD_MDF_ROOT	1	ルート ディレクトリ

SC_ModelDirectoryType

次の表に、SC_ModelDirectoryType のプロパティと列挙値を示します。

プロパティ	列挙値	説明
SCD_MDT_NONE	0	タイプは利用できません。
SCD_MDT_FILE	1	ファイル システム
SCD_MDT_MART	2	AllFusion Model Manager

SC_ModelObjectFlags

次の表に、SC_ModelObjectFlags のプロパティと列挙値を示します。

プロパティ	フラグ ビット	列挙値	説明
SCD_MOF_DONT_CARE		0	フラグは設定されません。
SCD_MOF_PERSISTENCE_UNIT	0	1	オブジェクトは永続ユニットです（モデルなど）。
SCD_MOF_USER_DEFINED	1	2	オブジェクトはユーザー定義です（ユーザー定義プロパティなど）。
SCD_MOF_ROOT	2	4	オブジェクトはルート オブジェクトです（モデルなど）。
SCD_MOF_TOOL	3	8	オブジェクトはツールによって管理されます。
SCD_MOF_DEFAULT	4	16	オブジェクトはツールによって作成され、削除不可です。

プロパティ	フラグビット	列挙値	説明
SCD_MOF_TRANSACTION	5	32	オブジェクトはトランザクション内で新規作成または更新され、コミットされていません。

SC_ModelPropertyFlags

次の表に、SC_ModelPropertyFlags のプロパティと列挙値を示します。

プロパティ	フラグビット	列挙値	説明
SCD_MPF_DONT_CARE		0	フラグは設定されません。
SCD_MPF_NULL	0	1	プロパティは NULL 値を持つか、または値なしです。
SCD_MPF_USER_DEFINED	1	2	プロパティはユーザー定義です。
SCD_MPF_SCALAR	2	4	プロパティはスカラです。
SCD_MPF_TOOL	3	8	プロパティはツールによって管理されます。
SCD_MPF_READ_ONLY	4	16	プロパティは読み取り専用です（ただし、AllFusion ERwin DM では使用されません）。
SCD_MPF_DERIVED	5	32	プロパティは継承値、計算値、または導出値です。
SCD_MPF_OPTIONAL	6	64	プロパティは任意であり、削除可能です。

SC_SessionFlags

次の表に、SC_SessionFlags のプロパティと列挙値を示します。

プロパティ	列挙値	説明
SCD_SF_NONE	0	セッションは、接続された永続ユニットに非排他的にアクセスします。他のセッションも、同一の永続ユニットに接続できます。
SCD_SF_EXCLUSIVE	1	セッションは、接続された永続ユニットに排他的にアクセスします。他のセッションは、同一の永続ユニットにアクセスできません。

SC_SessionLevel

次の表に、SC_SessionLevel のプロパティと列挙値を示します。

プロパティ	列挙値	説明
SCD_SL_NONE	-1	未使用
SCD_SL_M0	0	データ レベル アクセス
SCD_SL_M1	1	メタモデル アクセス

SC_ValueTypes

次の表に、SC_ValueTypes のプロパティと列挙値を示します。

プロパティ	列挙値	説明
SCVT_NULL	0	値なし
SCVT_I2	1	符号付き 16 ビット整数
SCVT_I4	2	符号付き 32 ビット整数
SCVT_UI1	3	符号なし 8 ビット整数。このタイプを文字データの保存に使用しないでください。
SCVT_R4	4	4 バイト浮動小数点実数
SCVT_R8	5	8 バイト浮動小数点実数
SCVT_BOOLEAN	6	ブール値
SCVT_CURRENCY	7	64 ビット通貨値
SCVT_IUNKNOWN	8	IUnknown インターフェイス ポインタ
SCVT_IDISPATCH	9	IDispatch インターフェイス ポインタ
SCVT_DATE	10	VARIANT_DATE 形式の日付値
SCVT_BSTR	11	文字列
SCVT_UI2	12	符号なし 16 ビット整数
SCVT_UI4	13	符号なし 32 ビット整数
SCVT_GUID	14	GUID
SCVT_OBJID	15	オフセットでオブジェクト識別子を含む文字列（VT_BSTR）
SCVT_BLOB	16	符号なし BYTE の SAFEARRAY
SCVT_DEFAULT	17	デフォルトの値タイプ
SCVT_I1	18	符号付き 1 バイト整数。このタイプを文字データの保存に使用しないでください。
SCVT_INT	19	マシン依存の符号付き整数

プロパティ	列挙値	説明
SCVT_UINT	20	マシン依存の符号なし整数
SCVT_RECT	21	4 つの整数からなる矩形配列（VT_ARRAY & VT_I2）
SCVT_POINT	22	2 つの整数からなるポイント配列（VT_ARRAY & VT_I2）
SCVT_I8	23	符号付き 64 ビット整数
SCVT_UI8	24	符号なし 64 ビット整数
SCVT_SIZE	25	2 つの整数からなるサイズ配列（VT_ARRAY & VT_I4）

プロパティ バッグのリファレンス

このセクションでは、Property Bag コンテナの内容について説明します。プロパティ バッグは、プロパティ配列のプレースホルダです。バッグの内容は、ソースのインターフェイスによって決定されます。

アプリケーション環境のプロパティ バッグ

このプロパティ バッグを使用して、Application Features セットにアクセスできます。PropertyBag 呼び出し時のパラメータによって、バッグの内容が決定されます。バッグの内容には、利用可能な 2 つの形式（ネイティブまたは文字列）から 1 つを指定できます。これには ISCApplicationEnvironment インターフェイスの PropertyBag プロパティの任意指定パラメータを使用します。

PropertyBag プロパティで、Category パラメータの Feature カテゴリは階層構造であり、ドット（.）を使用して機能のサブセットを定義します。たとえば、Application カテゴリは AllFusion ERwin DM のすべての機能を含むプロパティ バッグを表し、Application.API カテゴリは API に関連するサブセットを表します。

Category パラメータが設定されていない場合、Property Bag プロパティは利用可能なすべてのカテゴリの全機能を返します。

次の表に、利用可能な機能カテゴリを示し、各カテゴリの Property Bag プロパティを一覧表示します。

ISCApplicationEnvironment::PropertyBag

ISCApplicationEnvironment インターフェイスの PropertyBag 関数は、Category と Name で指定された 1 つ以上のプロパティ値を持つプロパティ バッグを追加します。

ISCApplicationEnvironment の PropertyBag 関数のシグネチャを示します:

```
ISCPropertyBag * PropertyBag(VARIANT Category, VARIANT Name, VARIANT AsString)
```

次の表に、ISCApplicationEnvironment の PropertyBag 関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
Category [省略可]	Empty	すべてのカテゴリの全機能セットを返します。
Category [省略可]	VT_BSTR – カテゴリ名	指定されたカテゴリの機能を返します。
Name [省略可]	Empty	選択されたカテゴリのすべてのプロパティを返します。
Name [省略可]	VT_BSTR – プロパティ名	指定された名前とカテゴリのプロパティを返します。
AsString [省略可]	Empty	プロパティ バッグのすべての値はネイティブ型になります。
AsString [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、プロパティ バッグのすべての値は文字列型になります。

空の文字列を含む Category パラメータ

次の表に、空の文字列を含む Category パラメータを示します。

プロパティ名	タイプ	説明
Categories	SAFEARRAY(BSTR)	利用可能なすべてのカテゴリの配列を返します。

Application カテゴリ

次の表で Application カテゴリについて説明します。AllFusion ERwin DM ツールに関連した機能が記述されます。

プロパティ名	タイプ	説明
Title	BSTR	AllFusion ERwin DM のタイトル
Version	BSTR	AllFusion ERwin DM のバージョン
Hosting Application	Long	■ 0 – API がサードパーティ アプリケーションによって制御されており、スタンドアロン モードである場合、0 を返します。 ■ 1 – AllFusion ERwin DM ユーザー インターフェイスがアクティブで、API がアドイン モードの場合、1 を返します。
Metadata Version	Long	現行バージョンの AllFusion ERwin DM のメタデータ値
EMX Metadata Class	SC_MODELTYPEID	EMX モデル セットのメタデータ クラス識別子
EM2 Metadata Class	SC_MODELTYPEID	EM2 モデル セットのメタデータ クラス識別子

Application.API カテゴリ

次の表で Application.API カテゴリについて説明します。API に関連した機能が記述されます。

プロパティ名	タイプ	説明
API Version	BSTR	API インターフェイスのバージョン
API Major Version Number	Long	API のメジャー バージョン番号
API Minor Version Number	Long	API のマイナー バージョン番号

Application.API.Features カテゴリ

次の表で Application.API.Features カテゴリについて説明します。主要な操作で API が提供するサポート レベルの一覧です。

プロパティ名	タイプ	説明
Undo	Long	操作を取り消す機能を表します。 ■ 0 – Undo をサポートしません。 ■ ゼロ以外 – Undo をサポートします。
Redo	Long	取り消した操作をやり直す機能を表します。 ■ 0 – Redo をサポートしません。 ■ ゼロ以外 – Redo をサポートします。
Change Logging	Long	クライアントと最後にシンクロした後のすべての変更をレポートする機能を表します。 ■ 0 – Change logging をサポートしません。 ■ ゼロ以外 – Change logging をサポートします。
Ownership Support	Long	オブジェクト所有権に対するアプリケーションのサポート レベルを問い合わせます。次のようなサポート レベルがあります。 ■ 0 – アプリケーションはオブジェクト所有権をサポートしません。 ■ 1 – アプリケーションは所有権をサポートします。所有権のメタ関連は循環を含みません。 ■ 2 – アプリケーションは所有権をサポートします。所有権のメタ関連は循環を含みます。

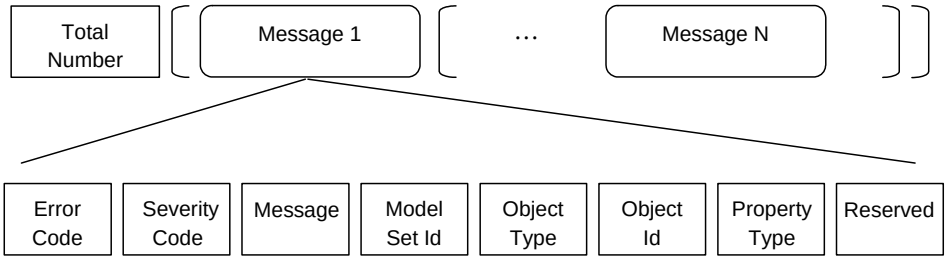
プロパティ名	タイプ	説明
Transactions	Long	トランザクション制御のサポート レベルを表します。次のようなサポート レベルがあります。 ■ 0 – トランザクションをサポートしません。 ■ 1 – Begin/End のみサポートします。ネストしたトランザクションはサポートしません。 ■ 2 – Begin、End、および Rollback をサポートします。ネストしたトランザクションはサポートしません。 ■ 3 – Begin、End、Rollback、およびネストしたトランザクションをサポートします。

Application.API.MessageLog カテゴリ

次の表で Application.API.MessageLog カテゴリについて説明します。API 操作時に記録された追加のメッセージにアクセスすることができます。

プロパティ名	タイプ	説明
Is Empty	ブール値	メッセージ ログが空でない場合、TRUE を返します。ログは API 操作の開始前にリセットされます。
Log	SAFEARRAY(VARIANT)	ログの内容を返します。

MessageLog カテゴリの Property Log は、VARIANT タイプを要素とする 1 次元の SAFEARRAY として構成されます。この配列は次のような構造になっています。



次の表に、配列の要素を示します。

メッセージ ログ要素	タイプ	説明
Total Number	Long	配列内のメッセージの総数。Log プロパティの要求時にメッセージが存在しない場合、この値はゼロになります。
Error Code	BSTR	メッセージ文字列の識別子

メッセージ ログ要素	タイプ	説明
Severity Code	Long	SC_MessageLogSeverityLevels の重大度コードは次のとおりです。 ■ SCD_ESL_NONE – 重大度コードは割り当てられません。 ■ SCD_ESL_INFORMATION – 情報メッセージ。 ■ SCD_ESL_WARNING – 警告メッセージ。 ■ ESD_ESL_ERROR – エラー メッセージ。
Message	BSTR	メッセージ テキスト
Model Set Id	SC_MODELSETID	メッセージに関連付けられたモデル セットの識別子。データが提供されない場合、要素は VARIANT タイプの VT_EMPTY になります。 Model Set 識別子を使用してモデル セットを指定する方法については、第 3 章「API タスク」の「モデルへのアクセス」および「モデル セットにアクセスする」セクションを参照してください。
Object Type	SC_CLSID	メッセージに関連付けられたモデル オブジェクトのクラス識別子。データが提供されない場合、要素は VARIANT タイプの VT_EMPTY になります。 Class 識別子の使用法とオブジェクト タイプの詳細については、第 3 章「API タスク」の「メタモデル情報へのアクセス」および付録 B「AllFusion ERwin DM メタモデル」を参照してください。
Object Id	SC_OBJID	メッセージに関連付けられたモデル オブジェクトの識別子。この識別子はモデル セットの範囲内で一意な値となります。データが提供されない場合、要素は VARIANT タイプの VT_EMPTY になります。 Object 識別子を使用して関連モデル オブジェクトにアクセスする方法の詳細については、第 3 章「API タスク」の「モデル オブジェクトへのアクセス」を参照してください。
Property Type	SC_CLSID	メッセージに関連付けられたプロパティのクラス識別子。データが提供されない場合、要素は VARIANT タイプの VT_EMPTY になります。 Class 識別子の使用法とプロパティ タイプの詳細については、第 3 章「API タスク」の「メタモデル情報へのアクセス」および付録 B「AllFusion ERwin DM メタモデル」を参照してください。
Reserved		常に VT_EMPTY に設定されます。

Application.Modeling カテゴリ

Application.Modeling カテゴリには、AllFusion ERwin DM のモデリング エンジンに関連した機能が記述されます。

プロパティ名	タイプ	説明
Model Property Value Facet IDs	SAFEARRAY(Long)	データは Long タイプを要素とする 1 次元の SAFEARRAY として構成されます。 要素は、モデル データで利用できるプロパティ値のファセット ID を表します。 要素の順序は、Model Property Value Facet Names と同一です。
Model Property Value Facet Names	SAFEARRAY(BSTR)	データは BSTR タイプを要素とする 1 次元の SAFEARRAY として構成されます。 要素は、モデル データで利用できるプロパティ値のファセット名を表します。 要素の順序は、Model Property Value Facet IDs と同一です。

次の表に、利用可能なプロパティ ファセットを一覧表示します。

プロパティ名	タイプ	説明
Hardened	5	継承によって値が変化しないことを示します。たとえば、子エンティティにおける外部キー属性の名前などです。
AutoCalculated	3	ツールによって値が自動計算されることを示します。たとえば、カーディナリティはデフォルトで自動計算されます。この場合、AutoCalculated ファセットは True に設定されます。

Application.Modeling.Physical カテゴリ

次の表で Application.Modeling.Physical カテゴリについて説明します。AllFusion ERwin DM の物理モデリングに関連した機能が記述されます。

プロパティ名	タイプ	説明
DBMS Brand And Version List	SAFEARRAY(Long)	データは Long タイプを要素とする 1 次元の SAFEARRAY として構成されます。 要素は 3 つのサブセットにグループ化されます。サブセットの最初のメンバは DBMS の種類を表す識別子です。2 つめのメンバはメジャー バージョン値で、最後のメンバはマイナー バージョン値です。

Application.Persistence カテゴリ

Application.Persistence カテゴリには、AllFusion ERwin DM の永続サポートに関連した機能が記述されます。このカテゴリにはプロパティはありません。

Application.Persistence.FileSystem カテゴリ

次の表で Application.Persistence.FileSystem カテゴリについて説明します。ファイル システムに関連した機能が記述されます。

プロパティ名	タイプ	説明
Current Directory	BSTR	現在のローカル ディレクトリの絶対パス

Application.Persistence.ModelMart

次の表で Application.Persistence.ModelMart カテゴリについて説明します。AllFusion Model Manager と共に動作する AllFusion ERwin DM の永続サポートに関連した機能が記述されます。

プロパティ名	タイプ	説明
Model Mart Connection Types	SAFEARRAY(BSTR)	AllFusion Model Manager がサポートするデータベース接続タイプの列挙値です。

モデル ディレクトリとモデル ディレクトリ ユニットのプロパティ バッグ

この Property Bag を使用して、Model Directory および Model Directory Unit オブジェクトのプロパティにアクセスできます。ISCModelDirectory インターフェイスおよび ISCModelDirectoryUnit インターフェイスの PropertyBag プロパティは、現在のプロパティ セットが含まれるバッグを追加します。これらのインターフェイスに共通のプロパティでは、ディレクトリ（読み取り専用でない場合）またはディレクトリ ユニットの属性を変更できます。プロパティ バッグの内容には、利用可能な 2 つの形式（ネイティブまたは文字列）から 1 つを指定できます。これには ISCModelDirectory および ISCModelDirectoryUnit の PropertyBag プロパティの任意指定パラメータを使用します。クライアントは、これら 2 つの形式のいずれかでバッグを追加できます。バッグの同一インスタンスで 2 つの形式を併用することもできます。

プロパティ バッグの追加時に、ディレクトリまたはディレクトリ ユニットに存在するすべてのプロパティを含める必要はありません。すべてのプロパティ データとプロパティ名は API によって検証され、すべてが許可または拒否されます。拒否されると、メソッドの呼び出しは失敗します。バッグに含まれるプロパティが、その時点で読み取り専用である場合（たとえば Locator プロパティ）、そのプロパティは無視され、プロパティ バッグのデータ検証には影響しません。

次の表に、Model Directory の Property Bag プロパティとデータタイプの一覧を示します。

プロパティ名	タイプ	読み取り専用	説明
Directory Name	BSTR	いいえ	パス情報を含まないディレクトリ名を返します。 新しい値を指定すると、ディレクトリ名が変更されます。 AllFusion Model Manager のルート ディレクトリでは、このプロパティ値はリポジトリ名を表します。このプロパティを使用して、リポジトリ名を変更することはできません。
Locator	BSTR	はい	絶対パスとパラメータを含むディレクトリの場所。AllFusion Model Manager では、パラメータにパスワード情報は含まれません。
Directory Path	BSTR	はい	ディレクトリの絶対パス。
Created By	BSTR	はい	ディレクトリを作成したユーザーID。AllFusion Model Manager では、AllFusion Model Manager のユーザー ID が使用されます。
Updated By	BSTR	はい	ディレクトリを更新したユーザーID。AllFusion Model Manager では、AllFusion Model Manager のユーザー ID が使用されます。
Created	SAFEARRAY(Long)	はい	ディレクトリの作成日時。次の順序で表された数値配列です。 ■ 秒（0～59） ■ 分（0～59） ■ 時（0～23） ■ 日（1～31） ■ 月（0～11; 1 月=0） ■ 年（現在の年） ■ 曜日（0～6; 日曜=0） ■ 年間通算日（0～365; 1 月 1 日=0）
Updated	SAFEARRAY(Long)	はい	ディレクトリの更新日時。次の順序で表された数値配列です。 ■ 秒（0～59） ■ 分（0～59） ■ 時（0～23） ■ 日（1～31） ■ 月（0～11; 1 月=0） ■ 年（現在の年） ■ 曜日（0～6; 日曜=0） ■ 年間通算日（0～365; 1 月 1 日=0）

プロパティ名	タイプ	読み取り専用	説明
Description	BSTR	いいえ	ディレクトリの説明。AllFusion Model Manager でのみ有効です。

次の表に、Model Directory Unit の Property Bag プロパティとデータタイプの一覧を示します。

プロパティ名	タイプ	読み取り専用	説明
Directory Unit Name	BSTR	いいえ	パス情報を含まないディレクトリ ユニット名を返します。 新しい値を指定すると、ディレクトリ ユニット名が変更されます。
Locator	BSTR	はい	絶対パスとパラメータを含むディレクトリ ユニットの場所。AllFusion Model Manager では、パラメータにパスワード情報は含まれません。
Directory Unit Path	BSTR	はい	ディレクトリ ユニットの絶対パス。
Created By	BSTR	はい	ユニットを作成したユーザーID。AllFusion Model Manager では、AllFusion Model Manager のユーザーID が使用されます。
Updated By	BSTR	はい	ユニットを更新したユーザーID。AllFusion Model Manager では、AllFusion Model Manager のユーザーID が使用されます。
Created	SAFEARRAY(Long)	はい	ディレクトリの作成日時。次の順序で表された数値配列です。 ■ 秒（0～59） ■ 分（0～59） ■ 時（0～23） ■ 日（1～31） ■ 月（0～11; 1 月=0） ■ 年（現在の年） ■ 曜日（0～6; 日曜=0） ■ 年間通算日（0～365; 1 月 1 日=0）

プロパティ名	タイプ	読み取り専用	説明
Updated	SAFEARRAY(Long)	はい	ディレクトリの更新日時。次の順序で表された数値配列です。 ■ 秒（0～59） ■ 分（0～59） ■ 時（0～23） ■ 日（1～31） ■ 月（0～11; 1 月=0） ■ 年（現在の年） ■ 曜日（0～6; 日曜=0） ■ 年間通算日（0～365; 1 月 1 日=0）
Description	BSTR	はい	ディレクトリの説明。AllFusion Model Manager でのみ有効です。
Model Type	Long	はい	ユニット モデルのタイプを取得します。モデル タイプは、論理、論理/物理、または物理のいずれかとなります。
Object Count	Long	はい	ユニット内のオブジェクトの総数を報告します。AllFusion Model Manager でのみ有効です。
Entity Count	Long	はい	ユニット内のエンティティ オブジェクトの総数を報告します。AllFusion Model Manager でのみ有効です。
Is Template	ブール値	はい	ユニット モデルがテンプレートの場合、True を返します。

永続ユニットと永続ユニット コレクションのプロパティ バッグ

このプロパティ バッグを使用して、永続ユニットのプロパティにアクセスできます。COM API の CoCreateInstance を呼び出すと、空のプロパティ バッグが取得されます。クライアントはバッグを追加し、ISCPersistenceUnitCollection インターフェイスの Create メソッドのパラメータとして送信します。代わりに、ISCPersistenceUnit の PropertyBag プロパティから、永続ユニット プロパティの現在の状態を取得することもできます。取得した値は確認および変更して、同じインターフェイスの PropertyBag プロパティに戻すことができます。バッグの内容には、利用可能な 2 つの形式（ネイティブまたは文字列）から 1 つを指定できます。これには ISCPersistenceUnit の PropertyBag プロパティの任意指定パラメータを使用します。クライアントは、これら 2 つの形式のいずれかでバッグを追加できます。バッグの同一インスタンスで 2 つの形式を併用することもできます。

バッグの追加時に、ユニットに存在するすべてのプロパティを含める必要はありません。すべてのプロパティ データとプロパティ名は API によって検証され、すべてが許可されるか、またはすべてが拒否されます。拒否されると、メソッドの呼び出しは失敗します。バッグに含まれるプロパティが、その時点で読み取り専用である場合（たとえば、モデルが作成されたときの AllFusion ERwin DM モデルのモデル タイプなど）、そのプロパティは無視され、バッグ データの検証に影響しません。

ISCPersistenceUnit::PropertyBag の引数（Get 関数）

PropertyBag（Get）関数のシグネチャを示します:

```
ISCPropertyBag * PropertyBag(VARIANT List, VARIANT AsString)
```

次の表に、PropertyBag（Get）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]	VT_BSTR – セミコロンで区切られたプロパティー覧	ユニット プロパティの一覧を指定します。一覧に指定されたプロパティのみが、返されるプロパティ バッグ内に配置されます。
List [省略可]	Empty	すべてのプロパティ セットを要求します。
AsString [省略可]	VT_BOOL – TRUE または FALSE	TRUE に設定すると、プロパティ バッグのすべての値が文字列型になります。デフォルトは FALSE で、すべての値はそれぞれのネイティブ型になります。
AsString [省略可]	Empty	プロパティ バッグのすべての値がネイティブ型になります。

ISCPersistenceUnit::PropertyBag の引数（Set 関数）

PropertyBag（Set）関数のシグネチャを示します:

```
void PropertyBag(VARIANT List, VARIANT AsString, ISCPropertyBag * propBag)
```

次の表に、PropertyBag（Set）関数の有効な引数を示します。

パラメータ	有効なタイプ/値	説明
List [省略可]		未使用
AsString [省略可]		未使用
propBag	ISCPropertyBag *	処理対象のユニット プロパティを含むプロパティ バッグのポインタ。

Property Bag Contents for Persistence Unit and Persistence Unit Collection

次の表に、AllFusion ERwin DM で認識される PropertyBag プロパティとデータタイプの一覧を示します。

プロパティ名	タイプ	読み取り専用	説明
Locator	BSTR	はい	永続ユニットの場所を返します（ファイル名など）。永続的な場所を持たないモデルでは利用できません（たとえば、一度も保存されていない新規モデルなど）。
Disposition	BSTR	はい	永続ユニットのディスポジション（読み取り専用など）を返します。
Persistence Unit Id	SC_MODELTYPEID	いいえ	永続ユニットの識別子を取得および設定します。 既存の永続ユニットに新しい識別子を割り当てることができます。この場合、古い識別子は永続ユニットの分岐ログに格納されます。詳細については、Branch Log プロパティの説明を参照してください。
Branch Log	SAFEARRAY (SC_MODELTYPEID)	作成後	永続ユニット識別子の分岐ログを取得および設定します。永続ユニットは識別子のログを保持します。 AllFusion ERwin DM は、永続ユニットの分岐ログを使用して、識別子による一致を拡張します。 API は最新の識別子のみを使用して、永続ユニットコレクション内を検索します。
Model Type	Long	作成後	永続ユニットのタイプを取得および設定します（たとえば、論理モデル、論理/物理モデル、および物理モデル）。永続ユニットの作成時に設定でき、そのプロパティは読み取り専用になります。 次の値を使用できます。 ■ 1 - Logical（論理モデルの指定）。値が指定されない場合のデフォルトです。 ■ 2 – Physical（物理モデルの指定）。 ■ 3 – Combined（論理/物理モデルの指定）。
Target Server Target Server Version Target Server Minor Version	Long	作成後	物理モデルと論理/物理モデルにおける対象データベースのプロパティを取得および設定します。永続ユニットの作成時に設定でき、そのプロパティは読み取り専用になります。 Target Server プロパティで利用可能な値については、次の表を参照してください。

プロパティ名	タイプ	読み取り専用	説明
Storage Format	Long	作成後	ストレージ形式を取得および設定します。モデルには Normal 値、モデル テンプレートには Template 値を指定します。永続ユニットの作成時に設定でき、そのプロパティは読み取り専用になります。 次の値を使用できます。 ■ 4012 – Normal（通常モデル）。値が指定されない場合のデフォルトです。 ■ 4016 – Template（テンプレート モデル）。
Active Model	ブール値	いいえ	永続ユニットが現在のモデルを表し、AllFusion ERwin DM ユーザー インターフェイスでアクティブである場合、TRUE となります。スタンドアロン モードの API では利用できません。
Hidden Model	ブール値	いいえ	永続ユニット データを表示するモデル ウィンドウが AllFusion ERwin DM ユーザー インターフェイスで非表示である場合、TRUE となります。スタンドアロン モードの API では利用できません。
Active Subject Area and Stored Display	SAFEARRAY(BSTR)	いいえ	アクティブなサブジェクト エリアおよびストアド ディスプレイのモデル オブジェクト名をレポートします。これは AllFusion ERwin DM の画面に表示されているサブジェクト エリアとストアド ディスプレイです。戻り値は、2 つの要素を持つセーフ配列です。第 1 要素はアクティブなサブジェクト エリアの名前、第 2 要素はアクティブなストアド ディスプレイの名前です。 サブジェクト エリアとストアド ディスプレイの新しい名前セットを指定すると、選択項目を変更できます。モデルが AllFusion ERwin DM ユーザー インターフェイスでアクティブであるか、既に開いている別のモデルである場合、この変更は即座に反映されます。 新規サブジェクト エリアの名前を指定する BSTR のみを使用して、選択項目を変更することもできます。指定されたサブジェクト エリアから、API は最初のストアド ディスプレイをアクティブにします。

Target Server プロパティは、3 つのメンバから構成されるベクトルです。ベクトルの最初のメンバは DBMS の種類を表す識別子です。2 つめのメンバはメジャー バージョン値で、最後のメンバはマイナー バージョン値です。

次の表に、Target Server プロパティで DBMS の種類を表す識別子の一覧を示します。また、識別子が文字列として表された識別名の一覧も示します。

DBMS の種類	DBMS の識別名	DBMS の識別 ID
Access	Access	1075859004
Advantage Ingres	Ingres	1075859007

DBMS の種類	DBMS の識別名	DBMS の識別 ID
DB2	DB2	1075858978
DB2 UDB	DB2 UDB	1075858977
FoxPro	FoxPro	1075859029
Informix	Informix	1075859006
iSeries (AS400)	iSeries	1075859019
ODBC	ODBC	1075859009
Oracle	Oracle	1075858979
Progress	Progress	1075859010
Red Brick	Red Brick	1075859012
SAS	SAS	1075859013
SQL Server	SQL Server	1075859016
Sybase	Sybase	1075859017
Teradata	Teradata	1075859018

セッションのプロパティ バッグ

この Property Bag は、ISCSession インターフェイスの BeginNamedTransaction 関数に追加情報を提供します。この関数の 2 番目の（任意指定の）引数として指定できます。バッグの内容には、利用可能な 2 つの形式（ネイティブまたは文字列）から 1 つを指定できます。クライアントは、これら 2 つの形式のいずれかでバッグを追加できます。バッグの同一インスタンスで 2 つの形式を併用することもできます。

送信時には、すべてのプロパティをバッグに含める必要はありません。すべてのプロパティデータとプロパティ名は API によって検証され、すべてが許可されるか、または拒否されます。拒否されると、メソッドの呼び出しは失敗します。

このトランザクション プロパティは、外部トランザクションの開始時に有効となり、トランザクションのスコープに制限されます。

次の表に、BeginNamedTransaction の PropertyBag プロパティとデータタイプの一覧を示します。

プロパティ名	タイプ	読み取り専用	説明
History Tracking	ブール値	いいえ	TRUE – トランザクションの間に生成されたすべての履歴情報が、API イベントとしてマークされます。このプロパティが指定されない場合、TRUE 値が設定されます。 FALSE – 標準の AllFusion ERwin DM メカニズムを使用して履歴が追跡されます。
History Description	BSTR	いいえ	History Tracking プロパティが TRUE の場合、履歴イベントの Description フィールドの内容が格納されます。

モデル ディレクトリおよび永続ユニットの場所とディスポジション

永続ストレージ機能における永続ユニットの場所とディスポジションは、Locator および Disposition プロパティで記述されます。この情報は API の一部のメソッドでは必須です。プロパティ バッグを使用してアクセスすることもできます。AllFusion ERwin DM モデルの永続ストレージの例は、ファイル システムと AllFusion Model Manager です。

Locatorプロパティ

次の表に、Locator プロパティでサポートされる構文を説明します。

構文	引数
[provider://]pathinfo[?param=value[;param=value]…n]	provider: 永続ストレージのタイプ。ファイル システムには erwin、AllFusion Model Manager リポジトリには mmart を指定します。省略すると、デフォルトで erwin が指定されます。 pathinfo: ストレージの場所を表すパス。ファイル パスまたは AllFusion Model Manager パスのいずれかを指定します。 param: パラメータ名またはキーワードのいずれかを指定します。 value: テキスト文字列。

ファイル システムの永続ストレージでは、param キーワードは定義されません。

provider パートに mmart を指定する際に使用できる param キーワードの一覧については、次の表を参照してください。

注: AllFusion Model Manager の Locator には、特別な設定があります。アプリケーションが 1 つ以上の AllFusion Model Manager リポジトリに接続済みである場合、Locator 文字列の param パートは省略することができます。この場合、Locator 文字列の param パートには一部の情報のみを指定するか、またはすべて省略することができます。これは、利用可能な一覧からどの接続を参照しているかが明らかな場合に限ります。

現時点では、AllFusion ERwin DM クライアントは、AllFusion Model Manager リポジトリとの間に 1 つの接続のみを開くことができます。したがって、接続を確立した後は、Locator 文字列の param パートをすべて省略してモデルのパス情報（pathinfo パート）のみを指定することができます。

provider パートに mmart を指定する際に使用できる param キーワードの一覧については、次の表を参照してください。

完全な名前	略語	説明
Server	SRV	AllFusion Model Manager サーバー。

完全な名前	略語	説明
Server Type	SRT	サーバー接続のタイプ。詳細については、「Application.Persistence.ModelMart」セクションを参照してください。
Trusted Connection	TRC	SQL Server バージョン 7 以降、および Oracle バージョン 8 以降。YES に設定すると、データベース ドライバで Windows 認証モードをログイン検証に使用します。UID および PSW キーワードは任意です。No に設定すると、データベースドライバでデータベースのユーザー名とパスワードをログイン検証に使用します。UID および PSW キーワードの指定は必須です。
User	UID	ログイン ユーザー名。Windows 認証を使用する場合は、UID を指定する必要はありません。
Password	PSW	ユーザーのログイン パスワード。ログイン パスワードが NULL であるか、または Windows 認証を使用する場合（Trusted Connection を YES に設定）、PSW を指定する必要はありません。
Submodel Name	SBN	アクセスするサブモデルの名前。
Library ID	LID	ライブラリ ID、モデル ID、およびサブモデル ID（いずれも任意）。指定すると、モデルを開く際のアクセス時間を短縮できます。
Model ID	MID	
Submodel ID	SID	

次の表に、Locator で param キーワードを使用するいくつかのシナリオを示します。最初の 2 つは、AllFusion Model Manager に保存されたモデルのプロバイダである mmart タイプを使用する例です。

シナリオ	説明
AllFusion Model Manager で SQL Server を使用	MyModel という名前のモデルが AllFusion Model Manager の MyMart の MyLib に保存されている場合、次のように指定します。 mmart://MyMart/MyLib/MyModel?SRV=MyServer;SRT=SQL Server 2000/2005 (using db-lib);UID=<Model Manager のユーザー ID>;PSW=<パスワード>
AllFusion Model Manager で Oracle を使用	MyModel という名前のモデルが AllFusion Model Manager の OracleHome の MyLib に保存されている場合（データベース名は指定せず、追加のスラッシュ（ / ）が必要です）、次のように指定します。 mmart:///MyLib/MyModel?SRV=OracleHome;SRT=Oracle Vers. 8.xx/9i/10g;UID=<Model Manager のユーザーID>;PSW=<パスワード>
ローカル ドライブ	mod.erwin という名前のモデルが C ドライブの models ディレクトリに保存されている場合、次のように指定できます。 C:\models\mod.erwin

Dispositionプロパティ

Disposition パラメータは、Locator パラメータで指定されたモデル データにアクセスする API に追加情報を提供します。

次の表に、Disposition プロパティによってサポートされる構文を説明します。

構文	引数
param=value[;param=value]…n	param: パラメータ名またはキーワードのいずれかを指定します。 value: Yes または No

次の表に、プロバイダに erwin タイプを指定するとき（ファイル システムに保存されたモデルを使用するなど）の Disposition param キーワード一覧を示します。

完全な名前	略語	説明
Read Only	RDO	ファイルの読み取り専用アクセスを要求します。Persistence Unit Collection の Add メソッドで利用できます。 永続ユニットへのフルアクセスは、このパラメータが指定されていない場合のみ可能です。
Override File	OVF	保存時に既存のファイルを上書きします。Persistence Unit の Save メソッドで利用できます。パラメータが指定されていない場合は上書きしません。

次の表に、AllFusion Model Manager に保存されたモデルのプロバイダである mmart タイプを使用する際の Disposition param キーワードの一覧を示します。

完全な名前	略語	説明
Read Only	RDO	AllFusion Model Manager ストレージのモデルはロックされません。Read Only 以外では、モデルは排他的にロックされます。
Override Locks	OVL	AllFusion Model Manager の開いているセッションが、あるユーザーに関連している場合でも、無視して続行します。
Resume Session	RSS	モデルがユーザーによって開かれており、一度も閉じられていない場合、AllFusion Model Manager はセッションを再開しようとします。
Override Model	OVM	Model Manager データベース内の既存のモデルを上書きします。Persistence Unit の Save メソッドで利用できます。パラメータが指定されていない場合は上書きしません。
Clear All Connections	CLC	他の Model Manager データベースに開いている接続をクリアして、新しいサーバーで続行します。

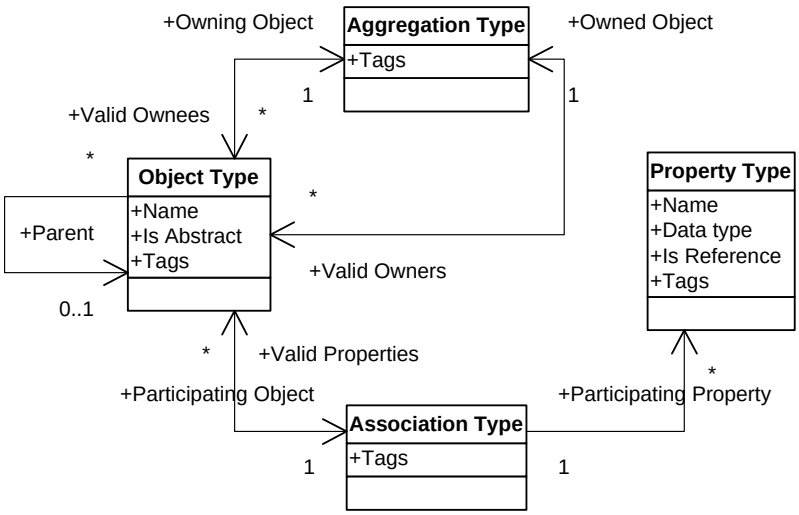
付録B: AllFusion ERwin DM メタモデル

この付録では、AllFusion ERwin DM メタモデルに関する情報を説明します。

本章には次のトピックがあります。

メタデータの構成

次のダイアグラムに、メタデータの構成を示します。

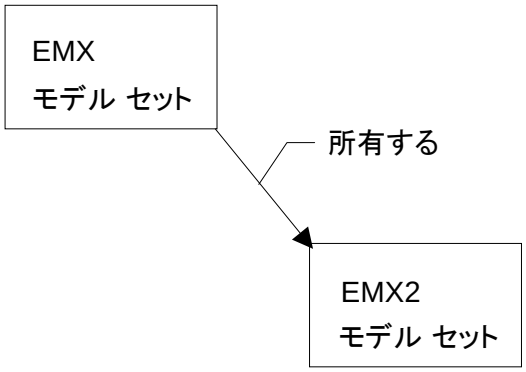


メタモデル要素

AllFusion ERwin DM のデータは、互いにリンクしたモデル セットのグループとして編成されます。モデル セットはツリー型の階層構造であり、階層のトップには 1 つのモデル セットが配置されます。

トップ モデル セットには、ほとんどのモデリング データが含まれています。API では、EMX という略語を使用してトップ モデル セットを表します。

EMX モデル セットには、EM2 という略語で表されるセカンダリ モデル セットがあります。EM2 には、フォワード エンジニアリングや完全比較など、AllFusion ERwin DM サービスのユーザー インターフェイス設定とユーザー オプションが含まれています。



API クライアントがモデル データにアクセスするには、セッションを作成してモデル セットに接続します。これには Session コンポーネントを使用します。

モデル セットには、さまざまなレベルのデータが含まれています。たとえば、エンティティ インスタンス、属性インスタンス、リレーションシップ インスタンスなど、アプリケーションが操作するデータが含まれています。

モデル セットにはメタデータも含まれています。メタデータは、アプリケーション データ内部に配置可能なオブジェクトおよびプロパティを記述したものです。

AllFusion ERwin DM のメタデータには、オブジェクトとプロパティのクラス、オブジェクト集約、およびプロパティ関連が含まれています。

オブジェクト クラス

モデル内に配置可能なオブジェクトのタイプを定義します。たとえば、エンティティ クラス、属性クラス、またはリレーションシップ クラスなどです。

プロパティ クラス

オブジェクトに設定できるプロパティのタイプを定義します。たとえば、Name プロパティ、Comment プロパティ、または Parent Domain Ref プロパティなどです。

オブジェクト集約

オブジェクト クラス間における所有権の関係を表します。たとえば、「モデルはエンティティを所有する」、「エンティティは属性を所有する」などです。

プロパティ関連

オブジェクト クラスによるプロパティの使用を定義します。たとえば、メタデータには、Name プロパティを持つすべてのオブジェクト クラスに対するプロパティ関連が含まれています。

メタデータ タグ

各メタデータ オブジェクトには 1 つ以上のタグが含まれています。タグとは、ある種の記述的なメタ情報を表すために使用されるメタデータ オブジェクトのプロパティです。たとえば、あるオブジェクト クラスについて「論理」、「物理」、「特定の対象 DBMS で有効」などの情報を表します。

オブジェクト集約のタグは、関連する実オブジェクト クラスに設定された同一のタグを上書きします。

プロパティ関連のタグは、関連するプロパティ クラスに設定された同一のタグを上書きします。

次の表に、EMX メタデータ タグの一部を示します。

タグの名前	データタイプ	説明
BitFieldValues … BitFieldValues10	String	ビット フィールド プロパティの有効値を記述します。記述された一覧の値を組み合わせて、プロパティの値に使用できます。 記述は次のようにグループ化されています。 {<value>|<string equivalent>|<internal>}
DBMS_Brands_And_Versions	Integer, vector	オブジェクトまたはプロパティ クラスが、特定の DBMS の物理モデリングで利用できる条件を定義します。IsPhysical タグは TRUE 値であることが前提です。 このタグが未設定の場合、クラスはすべての対象 DBMS で利用できます。ただし、IsPhysical タグが TRUE の場合に限ります。 このタグが NULL 値の場合、クラスはすべての DBMS で利用不可になります。 DBMS の識別 ID については、次の表で説明します。
DBMS_Is_Represented	Integer, vector	オブジェクトまたはプロパティ クラスが特定の DBMS の概念を表す条件を定義します。クラスの DBMS_Brands_And_Versions タグが有効であることが前提です。 このタグが未設定の場合、クラスはすべての対象 DBMS で利用できます。ただし、そのクラスで DBMS_Brands_And_Versions タグが有効な場合に限ります。 このタグが NULL 値の場合、クラスはすべての DBMS で利用不可になります。 DBMS の識別 ID については、次の表で説明します。

タグの名前	データタイプ	説明
DBMS_Is_Top_Level_Object	Integer, vector	オブジェクト クラスがトップ レベルとみなされる条件を定義します。たとえば、特定の DBMS に関連付けられた CREATE または DROP ステートメントを持つような場合などです。クラスの DBMS_Is_Represented タグが有効であることが前提です。 このタグが未設定の場合、クラスはすべての対象 DBMS で利用できます。ただし、そのクラスで DBMS_Is_Represented タグが有効な場合に限ります。 このタグが NULL 値の場合、クラスはすべての DBMS でトップ レベル オブジェクトではありません。 DBMS の識別 ID については、次の表で説明します。
EnumValues … EnumValues10	String	列挙値プロパティの有効値を記述します。記述された一覧の 1 つの値のみを、プロパティの値に使用できます。 記述は次のようにグループ化されています。 {<value>|<string equivalent>|<internal>}
IsFontOrColor	ブール値	モデル データの表示に関わるクラスで TRUE になります。
IsForDataMovement	ブール値	多次元モデリングやデータウェアハウス モデリングで利用できるオブジェクトまたはプロパティ クラスで TRUE になります。
IsGraphicData	ブール値	モデル データの表示に関わるクラスで TRUE になります。
IsLogical	ブール値	論理モデリングで利用できるオブジェクトまたはプロパティ クラスで TRUE になります。
IsPhysical	ブール値	物理モデリングで利用できるオブジェクトまたはプロパティ クラスで TRUE になります。
IsUserSettings	ブール値	AllFusion ERwin DM 機能のオプション保存に関わるクラスで TRUE になります。

DBMS 固有のタグ（DBMS_Brands_And_Versions、DBMS_Is_Represented、および DBMS_Is_Top_Level_Object など）はベクトルです。次の 3 つの要素をグループ化してデータを編成します。

第 1 要素

DBMS の識別 ID を指定します。

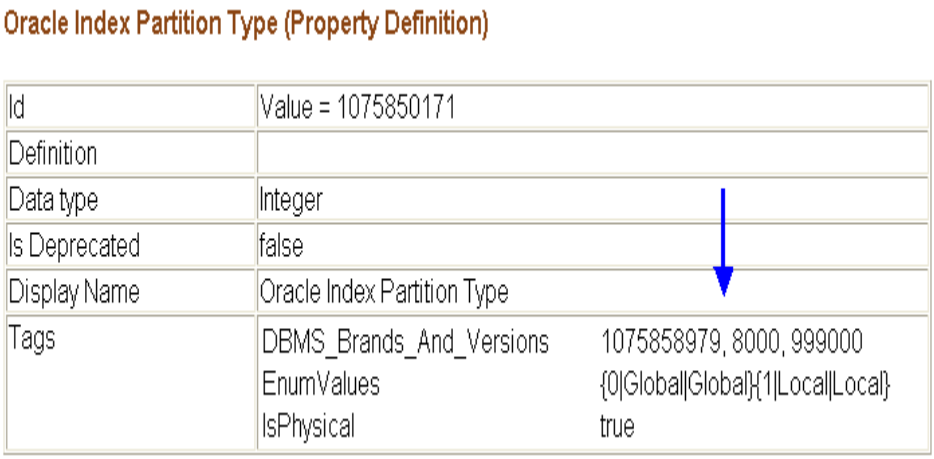
第 2 要素

DBMS の最小バージョン レベルを指定します。1000 を掛け合わせた値です。

第 3 要素

DBMS の最大バージョン レベルを指定します。1000 を掛け合わせた値です。値が 999000 の場合は最大レベルが無いことを示します。

次の図に、DBMS_Brands_And_Versions タグでのこれらの要素の例を示します。



第 1 要素である DBMS の識別 ID は Oracle を表し、1075858979 です。

第 2 要素である DBMS の最小バージョン レベル（1000 を掛け合わせた値）は 8000 です。この DBMS（Oracle）の最小 DBMS バージョン レベルが 8.0 であることを表します。

第 3 要素である DBMS の最大バージョン レベルは 999000 です。この DBMS の最大バージョン レベルが無いことを表します。

この例は、AllFusion ERwin DM で生成されたメタモデル文書から引用したものです。メタモデル文書の生成についての詳細は、「メタモデル文書」セクションを参照してください。

次の表に、DBMS の識別 ID 一覧を示します。

DBMS の種類	DBMS の識別 ID
Access	1075859004
Advantage Ingres	1075859007
DB2	1075858978
DB2 UDB	1075858977
FoxPro	1075859029
Informix	1075859006
iSeries (AS400)	1075859019
ODBC	1075859009
Oracle	1075858979
Progress	1075859010
Red Brick	1075859012
SAS	1075859013
SQL Server	1075859016

DBMS の種類	DBMS の識別 ID
Sybase	1075859017
Teradata	1075859018

抽象メタデータ オブジェクト

メタデータの編成には汎化が使用されます。これにより、汎化関連を使用して抽象オブジェクト クラスから特殊なオブジェクト クラスを導出することができます。特殊クラスは抽象としてマークすることができ、後続の特殊化のソースとして使用できます。

アプリケーション データの内部では、具体的な非抽象オブジェクト クラスのインスタンスのみが許可されます。AllFusion ERwin DM は汎化メカニズムを使用してメタデータをフラット化します。その際、抽象オブジェクト クラスの集約、関連、およびタグが具象オブジェクト クラス内に複製されます。

メタモデル クラス

一意なメタデータ クラスを使用して、モデル セットに含まれるメタデータのタイプが識別されます。

EMX クラス モデル セット

エンティティや属性など、モデル データの大部分が含まれています。クラス名は EMX です。クラス識別子は Application Environment コンポーネント、Application カテゴリ、EMX Metadata Class プロパティで定義された値です。

EM2 クラス モデル セット

AllFusion ERwin DM サービス（たとえばフォワード エンジニアリングや完全比較）のユーザー インターフェイス設定やユーザー オプションなどの追加データが格納されます。クラス名は EM2 です。クラス識別子は Application Environment コンポーネント、Application カテゴリ、EM2 Metadata Class プロパティで定義された値です。

メタデータのレポート

AllFusion ERwin DM では、メタデータの記述をさまざまな形式で利用することができます。

メタモデル文書

メタモデル文書の生成は、AllFusion ERwin DM の［ヘルプ］メニューから実行できます。

EMX モデル全体、または現在のモデルに対してメタモデル文書を生成できます。現在のモデルに対するメタモデル文書を生成する場合、追加のオプションを使用すると、現在のモデルの対象 DBMS に基づいて出力をフィルタできます。このオプションでユーザー定義プロパティを確認することもできます。

生成した文書には、次の定義が含まれます。

- オブジェクトの定義
- プロパティの定義
- 関連の定義（プロパティの所有権に対する）
- 集約の定義（オブジェクトの所有権に対する）

この文書のデフォルト出力は HTML 形式です。AllFusion ERwin DM では、この形式のテンプレートが用意されています。出力をカスタマイズした独自のテンプレートを作成することもできます。

オブジェクトおよびプロパティ クラスの名前は、定義の説明のタイトルとして表示されます。

Child Relationship (Object Definition)

オブジェクト タイプが、アップグレード対象の AllFusion ERwin DM 4.1.4 モデルに固有のタイプである場合、オブジェクト定義の Is Deprecated フィールドが TRUE になります。それ以外の場合、これらのオブジェクト タイプは利用できません。同様に、プロパティ タイプがアップグレード対象の AllFusion ERwin DM 4.1.4 モデルに固有のタイプである場合、プロパティ定義の Is Deprecated フィールドが TRUE になります。それ以外の場合、これらのプロパティ タイプは利用できません。

次の図は、Is Deprecated フィールドに TRUE が定義されたオブジェクト タイプの例です。

Child Relationship (Object Definition)

Id	1075839018
Long Id	{E7AB3CC0-47AD-4A10-8972-C0FB869EE7CB}+4020002a
Superclass	ERwin Object
Definition	
Is Abstract	false
Is Deprecated	TRUE ←
Display Name	Child Relationship
Collection Display Name	
Valid Owners	
Valid Properties	
Tags	Legacy_MM_Order 256

オブジェクト定義では、抽象オブジェクト タイプの Is Abstract フィールドは TRUE になります。これらのタイプが使用されるのは、メタデータの定義のみです。モデル データのオブジェクトには、抽象オブジェクト タイプはありません。

次の図は、Is Abstract フィールドに TRUE が定義されたオブジェクト タイプの例です。

Abstract Transform (Object Definition)

Id	1075839044
Long Id	{E7AB3CC0-47AD-4A10-8972-C0FB869EE7CB}+40200044
Superclass	ERwin Object
Definition	
Is Abstract	true ←
Is Deprecated	false
Display Name	Transform
Collection Display Name	Transforms
Valid Owners	
Valid Properties	
Tags	IsLogical true IsPhysical true Legacy_MM_Order 256 PhysicalNamePlural Transforms Physical_Name Transform

レポートには、すべての定義に一意な識別子が付きます。

Id	1075839044
Long Id →	{E7AB3CC0-47AD-4A10-8972-C0FB869EE7CB}+40200044

Long Id フィールドの値は、API のメタデータ オブジェクト識別子です。オブジェクト識別子とタイプ コードの情報については、第 2 章「API コンポーネント」の「オブジェクト識別子」セクションを参照してください。

メタモデル文書の生成についての詳細は、AllFusion ERwin DM のオンライン ヘルプを参照してください。

XMLスキーマ

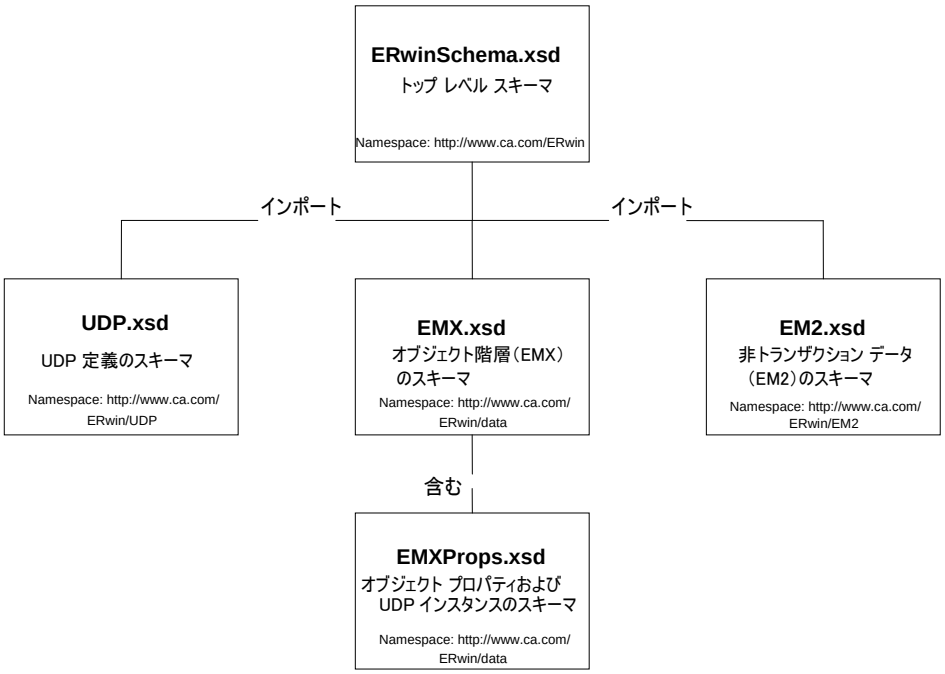
XML スキーマは、XML ファイルの構造と使用可能な要素を定義する文書です。XML スキーマを使用すると、XML ファイルの構文に誤りがなく、定義済みスキーマに適合していることを保証することができます。AllFusion ERwin DM では、XML ファイルを開く際に XML スキーマを使用してファイルを検証します。

AllFusion ERwin DM をインストールすると、XML ファイル検証に必要な XML スキーマ ファイル一式が Doc ディレクトリに保存されます。スキーマ ファイルの拡張子は .xsd です。次のような XML スキーマ ファイルが、AllFusion ERwin DM によって提供されます。

- ERwinSchema.xsd … トップ レベルのスキーマ ファイル
- UDP.xsd … UDP 定義のスキーマ ファイル
- EMX.xsd … オブジェクト階層のスキーマ ファイル
- EM2.xsd … 非トランザクション データのスキーマ ファイル
- EMXProps.xsd … オブジェクト プロパティおよび UDP インスタンスのスキーマ ファイル

XML スキーマには、モデル オブジェクトとプロパティ クラスの説明が含まれており、オブジェクト クラスに含まれるプロパティを定義します。EMX および EM2 クラスのスキーマ定義が提供されます。XML スキーマには廃止されたクラスは含まれていません。

次の図は、AllFusion ERwin DM で提供される 5 つの XML スキーマ ファイルのダイアグラムです。



Doc ディレクトリ下のスキーマ ファイルは、データベース固有ではなく、AllFusion ERwin DM メタモデル全体を表しています。このスキーマには、すべての有効な対象データベースで使用可能なオブジェクトおよびプロパティがすべて含まれています。データベース固有のスキーマが必要な場合は、Doc\DBMS_Schemas ディレクトリを参照してください。Doc\DBMS_Schemas ディレクトリ内に、AllFusion ERwin DM がサポートする対象データベースのフォルダがあります。データベース固有のスキーマ ファイルはこれらのフォルダに保存されています。特定の対象データベースで有効なオブジェクトとプロパティのみが含まれています。

注: AllFusion ERwin DM で XML ファイルを検証する際は、Doc ディレクトリにある XML スキーマが常に使用されます。データベース固有のスキーマは使用されません。データベース固有のスキーマは、文書化のために提供されており、サード パーティのツール開発者が、特定の対象データベースで有効なオブジェクトおよびプロパティを確認する際に役立ちます。外部の XML 検証ツールを使用すると、データベース固有のスキーマに対して XML ファイルを検証することができます。

ERwin Spy

ERwin Spy アプリケーションを使用すると、メタデータ情報を視覚化して、モデル固有の組み込みメタデータを表示することができます。AllFusion ERwin DM モデルの内部構造を確認することができるため、API を使用してアプリケーションを開発する際に役立ちます。ERwin Spy アプリケーションの詳しい情報と使い方については、第 2 章「API コンポーネント」の「ERwin Spy」セクションを参照してください。

AllFusion ERwin DM バージョン 4.1.4 からの変換

以下のセクションでは、バージョン 4.1.4 と r7.2 の間でのメタデータの変更点を説明します。これらのセクションで説明されるファイルは、AllFusion ERwin DM インストール ディレクトリ下の Doc ディレクトリに保存されています。

ユーザー定義プロパティ（UDP）

r7.2 では、UDP を定義するのに UDP Definition オブジェクト タイプを使用しません。UDP 定義はメタデータの一部です。新規 UDP を定義するには、新規 Property Type オブジェクトを EMX モデル セットのメタデータに追加する必要があります。

詳細については、第 3 章「API タスク」の「ユーザー定義プロパティを作成する」セクションを参照してください。

タイプ名の変更点

次の表に示すファイルには、名前が変更されたオブジェクトおよびプロパティのタイプのみが含まれています。これらのモデル データにおける役割は変更されていません。また、多数のオブジェクトとプロパティが廃止され、新しいオブジェクトおよびプロパティのタイプに置き換えられました。新しく追加されたり置き換えられたタイプは、次の「追加および廃止されたメタデータ」セクションの表に示すファイルに含まれています。

ファイル名	説明
Api Appendix B.LegacyApiNames.csv	名前が変更されたオブジェクトおよびプロパティのタイプの一覧です。各行には、r7.2 での新しい名前を示すカラムと、バージョン 4.1.4 での古い名前を示すカラム（「Legacy_ApiName= "名前"」という形式）があります。
Api Appendix B.EMX Renamed Objects.csv	名前が変更されたオブジェクト タイプの一覧です。
Api Appendix B.EMX Renamed Properties.csv	名前が変更されたプロパティ タイプの一覧です。

注: Api Appendix B. EMX Renamed Objects.csv と Api Appendix B.EMX Renamed Properties.csv は、名前が変更されたオブジェクトおよびプロパティのタイプの一覧をそれぞれ別に示したファイルです。

追加および廃止されたメタデータ

次の表に、メタデータの変更点が記載されたファイルを示します。

ファイル名	説明
Api Appendix B.EMX Object changes.csv	EMX メタモデルで追加および廃止されたオブジェクト タイプの一覧です。
Api Appendix B.EMX Property changes.csv	EMX メタモデルで追加および廃止されたプロパティ タイプの一覧です。
Api Appendix B.EMX Property usage changes.csv	EMX メタモデルへのプロパティ割り当ての変更履歴です。
Api Appendix B.EM2 Object changes.csv	EM2 メタモデルで追加および廃止されたオブジェクト タイプの一覧です。
Api Appendix B.EM2 Property changes.csv	EM2 メタモデルで追加および廃止されたプロパティ タイプの一覧です。

データタイプの変更点

Api Appendix B.EMX Property datatype changes.csv には、データタイプが変更されたプロパティの一覧が含まれています。AllFusion ERwin DM r7.2 は、バージョン 4.1.4 と同じような方法でプロパティを使用します。ただし、プロパティのデータタイプは、r7.2 で追加および拡張された点に従って変更されています。

索引

A

C

E

I

L

M

N

O

P

R

S

U

V

あ

か

さ

た

は

ま

ら